Konstruktion betriebswirtschaftlicher Standard-Anwendungssysteme aus Anwendungselementen

Vom Fachbereich Rechts- und Wirtschaftswissenschaften
der Technischen Universität Darmstadt genehmigte

Dissertation

zur Erlangung des akademischen Grades
Doctor rerum politicarum (Dr. rer. pol.)

vorgelegt von
Diplom-Informationswissenschaftler Klaus-Peter Lang
aus Lauffen am Neckar

Referent: Prof. Dr. Erich Ortner
Korreferent: Prof. Dr. Hubert Österle

Tag der Einreichung: 02. November 2004
Tag der Disputation: 01. Dezember 2005

Darmstadt 2006
D 17

Zusammenfassung

Zielsetzung der vorliegenden Arbeit ist die Analyse bewährter klassischer Ingenieurbereiche und die Untersuchung inwieweit ihre Methoden und Verfahren auf die noch jüngere Anwendungssystemkonstruktion übertragbar sind. Der Vergleich führt zu nachfolgendem Ergebnis.

Gerade die streng methodische Vorgehensweise für technische Konstruktionsprojekte und die ausgereiften Problemlösungsverfahren für den Entwurf komplexer Systeme sind attraktive Kandidaten für einen Einsatz der Erkenntnisse beim Entwurf umfangreicher betriebswirtschaftlicher Anwendungssysteme. Damit die Vorteile der ingenieurgemäßen Konstruktion deutlich erkennbar sind, wird die Entwicklung einer Konstruktionsmethode für komplexe Softwaresysteme (integrierte betriebswirtschaftliche Standard-Anwendungssysteme) in Angriff genommen. Dafür sind einerseits die Rahmenbedingungen in der Anwendungssystem-Architektur und im Vorgehen der Anforderungsanalyse, des Entwicklungs- und Einführungsprozesses zu bestimmen. Andererseits müssen die Methodenaspekte technischer Konstruktionslehre in einer sinnvollen Auswahl und einer geeigneten Form auf die Konstruktion von Anwendungssystemen angewandt werden.

Dies führt in Kapitel 2 dazu, dass für die Übertragung der klassischen Konstruktionsmethodik auf die Anwendungssystemkonstruktion ein Ordnungssystem für Anwendungskomponenten vorgeschlagen wird. Dieses Ordnungssystem besteht aus einem Spezifikationsrahmen (Sachmerkmalleisten), einem Komponenten-Katalog mit entsprechendem Konstruktionsverfahren (morphologische Matrix) und einem Baukastensystem (Repository-System).

Für das Herausarbeiten der Rahmenbedingungen in der Anwendungssystem-Architektur ist eine historische Entwicklung der Konstruktions- und Architekturmethoden für Anwendungssysteme dargestellt. Das Ergebnis dieser Untersuchung zeigt, dass erst eine serviceorientierte Architektur ein „Denken in Baugruppen“ soweit unterstützt, dass damit eine Business Services-Plattform für eine baukastenorientierte Herstellung von Lösungskomponenten möglich wird. Allerdings ist damit die Frage nach der Normierung und inhaltlichen Definition von Bauteilen noch keinesfalls beantwortet. Dafür ist ein auf den Anwendungsbereich fokussiertes sprachbasiertes Vorgehensmodell für die Anforderungsanalyse, die Entwicklung und die

Einführung notwendig. Eine Beschreibung der dafür geeigneten methodischen Grundlagen vollendet den analytischen Bereich der Arbeit.

Aufbauend auf die methodischen Grundlagen der technischen Konstruktion und mit Blick auf eine Business Services-Plattform sowie unter konsequentem Einsatz der Vorgehensweise sprachbasierter Informatik wird in Kapitel 3 die „Methode der semantischen Komposition“ systematisch entwickelt und erläutert. Dazu wird der Begriff „Anwendungselement“ als Softwarekomponente zur rechnerunterstützten Informationsverarbeitung im Kontext betrieblicher Aufgaben eingeführt. Dieses innovative Verständnis von aufgabenorientierten Anwendungssystembauteilen begründet die Möglichkeit, sowohl die Anwendungsprogramm-Logik zu gliedern als auch ein Baukastensystem für Konstruktionselemente aufzubauen.

Anhand eines Anwendungsbeispiels werden verschiedene Aspekte der Methode schrittweise erläutert. Dies sind die Anwendung der Konstruktionsprinzipien Abstraktion/ Konkretion und Komposition/Partition auf den Entwurf und die Herstellung der Bausteine eines Gesamtsystems mit all ihren Designprinzipien (Integrationsaspekte, Kohärenz und Kopplung, Orthogonalität und Faktorenanalyse).

In Kapitel 4 der Arbeit werden die Erfahrungen mit einem Forschungsprototypen eines Komponentenverwaltungssystems in die Diskussion eingebracht. Mit diesem Werkzeug wird gezeigt, wie Anwendungen aus Anwendungselementen aufgebaut werden können und dass durch Auswahl und Komposition eine kundenindividuelle Einführung möglich ist.

Abstract

The goal of this study is an analysis of established classical engineering science and an investigation of how its methodologies and procedures can be used for the comparatively young science of application software system construction. The comparison leads to the following conclusions:

The strict methodical procedure and the well-tried problem solving approaches from classical engineering for the design of highly complex systems are attractive candidates for applying to the design challenges faced when creating substantial business applications software. The advantages of engineering design is demonstrated by developing and presenting a complete methodology for the construction of complex software systems, in this case integrated standard business software. To achieve this, the general conditions about application architecture and procedures for analysis, development and implementation must be defined on the one hand. On the other hand, those aspects of engineering construction methodology that are best applied to application system construction need to be determined. This leads to the insight, presented in chapter 2, that a classification system for application components is necessary to support the whole system lifecycle. The classification system consists of a specification framework, a component catalogue with a corresponding construction procedure (morphological matrix) and finally the concept of modular construction systems (repository system).

A historical overview about architectural methodologies shows that only service-oriented architectures support the “building block principles” in a way which is needed to build modular construction systems for application components. This is the technical approach, but the more important question is, how to design and standardize the building blocks. This question is answered by an investigation of a natural language-based process model, which considers business domains for requirements engineering, development and implementation, and which completes the analytical part of the study.

Based on the above findings from engineering construction and keeping in mind the service-oriented application architecture and the consistent use of natural language-based application development principles it is possible to develop and present a holistic “methodology for semantical composition” of application components, as shown in chapter 3. The main pillar of the methodology is the new concept of “application

elements” in the sense of software components for computer-based information processing in the context of corporate business tasks. This innovative use of business task-oriented application building blocks enables one to subdivide the application program logic on the one hand and to build a modular construction system on the other hand.

Various facets of the methodology are described step by step using a consistent application example. The construction principles “abstraction and concretion” as well as “composition and partition” for designing and developing building blocks for an entire application system are demonstrated, and also specific component design aspects like integration, coherence and coupling, orthogonality and factor analysis are considered and explained.

Chapter 4 presents experiences with a software prototype component management system. The results show how configuration of business applications based on application elements could look like. Moreover, this research tool has proven on a small scale that customer requirements in the application implementation phase can be fulfilled by solely using component selection and composition.

Inhaltsverzeichnis ..... **I**

Abbildungsverzeichnis ..... **IV**

Tabellenverzeichnis ..... **VIII**

Abkürzungsverzeichnis ..... **IX**

Kapitel 1 Einleitung **1**

Kapitel 2 Konstruktionslehre **4**

2.1 Technische Konstruktionslehre **7**

2.1.1 Methodisches Vorgehen 9

2.1.2 Problemlösung 12

2.1.2.1 Modularisierung 14

2.1.2.2 Aufsteigendes Vorgehen 16

2.1.2.3 Adaptives Vorgehen 19

2.1.2.4 Rückgriff auf Vorhandenes 19

2.1.2.5 Zusammenfassung Problemlösung 24

2.1.3 Systematische Lösungsentwicklung 24

2.1.4 Baukastensysteme, Varianten, Baureihen, Typengruppen 28

2.1.5 Sachmerkmalleisten 33

2.1.6 Konstruktionsarten 35

2.1.7 Zusammenfassung 36

2.2 Konstruktionstheorie für betriebswirtschaftliche Standard-Anwendungssysteme **38**

2.2.1 Zusammenfassung 49

2.3 Sprachbasierte Anwendungsentwicklung **50**

2.4 Diskussion **54**

Kapitel 3 Aufbau und Inhalt der Methode der semantischen Komposition von Anwendungssystemen 58

3.1 Unterteilung der Komponenten nach Abstraktionsebenen 59
3.2 Komponentenorientierung auf Herstellungsebene 63
3.2.1 Herstellung von Anwendungselementen 63
3.2.2 Herstellung von Creatorelementen 65
3.3 Komponentenorientierung auf Anwendungsbereichsebene 67
3.3.1 Aufgabenorientierung 67
3.3.2 Einflüsse auf die semantische Komposition von Anwendungssystemen ... 71
3.3.3 Beispiel-Anwendungselemente 74
3.3.4 Konstruktionsprinzipien 79
3.3.4.1 Abstraktion - Konkretion 80
3.3.4.2 Komposition - Partition 81
3.3.5 Arbeitsteilung bei einer komponentenorientierten Anwendungsentwicklung 90
3.3.6 Wiederverwendung durch Auswahl versus Wiederverwendung durch Anpassung 95
3.3.7 Anwendungselemente-Baukastensysteme 101
3.3.8 Kommunikation zwischen Anwendungselementen 106
3.3.8.1 Anforderungen der Kombination und Variation von Anwendungs-elementen und Baugruppen an ihre Schnittstellengestaltung 111
3.3.8.2 Realisierungsmöglichkeiten der Komponenten-Kommunikation 119
3.4 Aufbau eines Ordnungssystems für Anwendungselemente 126
3.4.1 Technologieunabhängige Beschreibung von Anwendungselementen 126
3.4.2 Sachmerkmalleisten für Anwendungselemente 130
3.4.3 Konstruktionskatalog mit polyhierarchischer Gliederung 143
3.4.4 Semantische Relationen zwischen Anwendungselementen 152
3.4.5 Wissensbasiertes Konfigurieren 160
3.4.6 Terminologiebasierter Aufbau des Ordnungssystems 169
3.5 Zusammenfassung der Methode der semantischen Komposition 184

Kapitel 4 Anwendung der Methode der semantischen Komposition auf die Konstruktion von Standard-Anwendungssystemen 188

4.1 Terminologiebasiertes Komponenten-Management-System (TKMS) 189

4.1.1 Aufbau des Systems 189

4.1.2 TKMS Beispielanwendung 193

4.1.3 Übergang zwischen Anwendungselement und der Implementierung seines Creatorelementes 197

4.1.4 Datenablage für das Anwendungselemente-Beispiel in Ordnungssystem und Fachterminologie von TKMS 203

4.2 Durchführung der Variantenkonstruktion 206

4.3 Vereinfachte Variantenkonstruktion für Mittelstandsunternehmen 212

Kapitel 5 Schlussbetrachtungen 214

Literaturverzeichnis 219

Lebenslauf 233

Abbildungsverzeichnis

Abbildung 1: Magisches Viereck der Zielkonflikte in der Anwendungssystementwicklung 5

Abbildung 2: Übersicht der Einflussfaktoren auf die Konstruktionsmethodik 8

Abbildung 3: Generelles Vorgehensmodell für die technische Entwicklung und Konstruktion 10

Abbildung 4: Problempartition und Lösungskomposition 15

Abbildung 5: Strukturelle Darstellung von Abstraktion/Konkretion und Komposition/ Partition 18

Abbildung 6: Aufbau eines Konstruktionskataloges mit "eindimensionalem" Gliederungsteil 22

Abbildung 7: Beispiel eines Konstruktionskataloges: Welle-Nabe-Verbindungen, Übersichtskatalog 23

Abbildung 8: Beispiel eines Morphologischen Kastens für die Synthese eines Verschlussmechanismus eines Autoklavdeckels 25

Abbildung 9: Beispiel eines Baukastensystems für Radialgleitlager 29

Abbildung 10: Bausteinarten für Baukasten- und Mischsysteme 30

Abbildung 11: Matrix von Glühlampeneigenschaften und Sortimentangebot 32

Abbildung 12: Begriffsvarianten von „Variante“ und „Baureihe“ 33

Abbildung 13: Beispiel einer Sachmerkmalleiste für Passschrauben nach 34

Abbildung 14: Multipfad-Vorgehensmodell für die Anwendungssystem-Entwicklung 53

Abbildung 15: Szenario für die komponentenorientierte Anwendungsentwicklung.....62

Abbildung 16: Vorgehensweise für das Einrichten eines Anwendungssystems.....72

Abbildung 17: Übersicht der Einflüsse auf eine komponentenorientierte Anwendungssystementwicklung.....74

Abbildung 18: Variabilitätsfaktoren des Anwendungselementes *Kundenbestellung*.....76

Abbildung 19: Anwendungselemente als Kommunikationsbasis.....94

Abbildung 20: Wiederverwendungsarten.....96

Abbildung 21: Parameter versus Variantenauswahl.....97

Abbildung 22: Erzeugung von Baugruppen.....98

Abbildung 23: Aktionsbereiche für die Variabilität komponentenorientierter Anwendungssysteme.....107

Abbildung 24: Kommunikationsmöglichkeiten des Anwendungselementes AE1.....113

Abbildung 25: Baugruppen-Kommunikation und Kommunikationskonflikte.....116

Abbildung 26: Sachmerkmalleiste für Anwendungselemente.....131

Abbildung 27: Sachmerkmalleiste *Kundenbestellung*.....134

Abbildung 28: Gegenstandsmuster der *Kundenbestellung für Lagerartikel*.....134

Abbildung 29: Beschreibungsinformationen für den Kompositionsprozess.....140

Abbildung 30: Beispiel für eine polyhierarchische Gliederungsstruktur.....144

Abbildung 31: Abstraktionsbeziehungen außerhalb der Gliederung.....146

Abbildung 32: Strukturelemente und strukturelle Beziehungen innerhalb der polyhierarchischen Gliederung 147

Abbildung 33: Architektur eines Expertensystems für die Anwendungselemente-Konfiguration 161

Abbildung 34: Ausschnitt eines Beziehungsgraphen für Thesaurusbeziehungen 173

Abbildung 35: Aussagentransformation 176

Abbildung 36: Aussagentransformation der Methode der semantischen Komposition 179

Abbildung 37: Terminologische Basis der Methode der semantischen Komposition..182

Abbildung 38: Übersicht der Methode der semantischen Komposition.....187

Abbildung 39: Datenmodell des Terminologiebasierten Komponenten-Management-Systems dargestellt in der Objekttypenmethode von Ortner und Söllner.....190

Abbildung 40: Benutzeroberfläche des Anwendungselementes *Streckenbestellung* .194

Abbildung 41: Benutzeroberfläche des Anwendungselementes *Lagerbestellung* 196

Abbildung 42: Benutzeroberfläche des Anwendungselementes *Lagerbestellung* mit geöffnetem Dialogfenster für Lagerinformation.....197

Abbildung 43: UML-Klassendiagramm für Creatorelement Kundenbestellung.Lager.199

Abbildung 44: Datenbankzugrif für das Java Objekt *KundenbestellPosition* 200

Abbildung 45: Vereinfachter Konstruktionsprozess einer Anwendung aus Komponenten.....208

Abbildung 46: Benutzerspezifischer Arbeitsplatz 210

Abbildung 47: Lösungsauswahl im Anwendungselemente Repository.....211

Tabellenverzeichnis

Tabelle 1: Anforderungen an die Herstellung und den Einsatz von Anwendungssystemen 6

Tabelle 2: Schema zur Erfassung entwicklungsrelevanter Aussagen im Fachentwurf 52

Tabelle 3: Weiterentwicklung von Baukastensystemen 71

Tabelle 4: Beispiel für zwei digitalisierte Sachmerkmalleisten 87

Tabelle 5: Gegenüberstellung von Parametrisierung und Wiederverwendung durch Auswahl 100

Tabelle 6: Gegenstandsmuster der *Kundenbestellung für Lagerartikel* 133

Tabelle 7: Beispiele für Komplexmerkmale 135

Tabelle 8: Beziehungsarten der semantischen Relationen 153

Tabelle 9: Eigenschaften der semantischen Relationen 154

Tabelle 10: Übersicht der Implikationsbeziehungen 156

Tabelle 11: Matrix der möglichen Beziehungselemente-Kombinationen im Ordnungssystem 157

Tabelle 12: Matrix des eingeschränkten Beziehungsarteneinsatzes in den Beziehungselemente- Kombinationen 158

Tabelle 13: Beispiel eines Lexikoneintrags 171

Tabelle 14: Aussagetypen mit Schlüsselworten 180

Tabelle 15: Gegenstandsmuster für die *Lagerbestellung* in *TKMS* 198

Abkürzungsverzeichnis

A2A:	Application to Application
AE:	Anwendungselement
B2B:	Business to Business
BAPI:	Business Application Programming Interface
BSP:	Business Services-Plattform
COM+:	Enhanced Component Object Model
CORBA:	Common Object Request Broker Architecture
CTI:	Computer Telephony Integration
DCOM:	Distributed Component Object Model
DGD:	Deutsche Gesellschaft für Dokumentation
EAI:	Enterprise Application Integration
EJB:	Enterprise Java Bean
ESA:	Enterprise Services Architecture
HTTP:	Hypertext Transfer Protocol
IDL:	Interface Definition Language
IIOP:	Internet Inter-ORB Protocol
J2EE:	Java 2 Platform, Enterprise Edition
JNDI:	Java Naming Directory Interface

KI:	Künstliche Intelligenz
ORB:	Object Request Broker
OTM:	Objekttypenmethode
RMI:	Remote Method Invocation
SML:	Sachmerkmalleiste
SOA:	Service Oriented Architecture
SOAP:	Simple Object Access Protocol
TKMS:	Terminologiebasiertes Komponenten-Management-System
UDDI:	Universal Description, Discovery and Integration
UML:	Unified Modelling Language
URI:	Uniform Resource Identifier
WSDL:	Web Service Description Language
XML:	Extensible Markup Language

Kapitel 1 Einleitung

Ein wesentliches Unterscheidungsmerkmal zwischen Softwaresystemen im Allgemeinen und betriebswirtschaftlichen Anwendungssystemen im Besonderen ist das hohe Anspruchsniveau an die semantische Integration und Aufgabenorientierung entlang jeder denkbaren Wertschöpfungskette in einem Unternehmen bzw. einer Organisation. Dinge und Sachverhalte werden als semantisch bezeichnet, wenn sie in ihrer Bedeutung inhaltlich, fachlich und anwendungsbereichsbezogen definiert sind. Im Gegensatz zu syntaktischen Beschreibungen, die formal, technisch und strukturorientiert ausgeprägt werden.

In der vorliegenden Arbeit werden ausschließlich Anwendungssysteme zur Bearbeitung von betriebswirtschaftlichen Prozessen, Funktionen und Methoden betrachtet. Dabei wird ausdrücklich der Problematik der semantischen Integration und Aufgabenorientierung Rechnung getragen. Mit dieser Zielsetzung wird eine zweite Unterscheidung deutlich, die es zu berücksichtigen gilt: Individualsoftware im Vergleich zu Standardsoftware.

Individuell für ein konkretes Unternehmen entwickelte Anwendungsprogramm-Logik unterscheidet sich wesentlich in ihrer Variabilität und somit Komplexität von der

Anwendungsprogramm-Logik von Standard-Anwendungssystemen. Individualsysteme treffen konkrete Entscheidungen über den Daten- und Kontrollfluss für genau diese Fachabteilung oder dieses Unternehmen. Standard-Anwendungssysteme müssen eine semantische Integration über die Prozess- und Funktionsgrenzen hinweg, mit der maximalen Variabilität aller möglichen Kundeneinsatz-Szenarien für das System, sicherstellen. Dadurch sind sie allgemeiner (generischer) und folglich komplexer zu konstruieren. Herstellung von Individualsoftware bedarf weniger einer anspruchsvollen Konstruktionsmethodik, sondern vielmehr einer optimierten Werkzeugunterstützung. Modellgetriebene „Rapid Development"-Werkzeuge mit entsprechenden „Re-factoring"-Eigenschaften, die eine inkrementelle und zyklische (round-trip) Vorgehensweise unterstützen, stehen derzeit in der Gunst der Kunden für individuelle Software Entwicklungsprojekte. Die Entwicklung von Individualsoftware wird in der vorliegenden Arbeit nicht weiter betrachtet.

Im Bereich der Entwicklung von Standard-Anwendungssystemen ist, bedingt durch die höhere intrinsische Komplexität, der Einsatz ingenieurtechnischer Konstruktionsmethoden unerlässlich. Nur dadurch kann eine Beherrschbarkeit der Flexibilität in der Programm-Logik und Architektur gleichzeitig dem Anspruch einer einfachen Benutzbarkeit und Verstehbarkeit durch den End-Anwender gerecht werden. Hinzu kommt, dass die Herstellungsprozesse für Anwendungssysteme, wie in der Fertigungsindustrie, mehrstufigen Fertigungsprozessen gerecht werden müssen. Dies kann nur durch methodisch angepasste Verfahren gewährleistet werden. Um der wichtigen Erkenntnis Rechnung zu tragen, dass die Grundlage für eine erfolgreiche Konstruktionsmethodik im betriebswirtschaftlichen Integrationsumfeld eine semantische Kombinierbarkeit von semantisch begründeten Bauelementen voraussetzt, wurde im Rahmen der vorliegenden Arbeit die ***Methode der semantischen Komposition*** entwickelt. Die Ergebnisse zeigen, wie die ***Methode der semantischen Komposition*** aufbauend auf den Vorbildern aus der technischen Konstruktionslehre als katalogbasierte, baukastenorientierte Methode zur Variantenkonstruktion entwickelt wurde, die im Speziellen das Modularisierungsproblem von Standard-Anwendungssystemen mittels semantischer Bauelemente, den so genannten „Anwendungselementen", gelöst hat.

Die vorliegende Arbeit vermittelt in Kapitel 2 eine Übersicht über die Konstruktionslehre in technischen Wissenschaftsbereichen, in der Softwareentwicklung für betriebswirtschaftliche Anwendungssysteme und in der terminologiebasierten Anwendungssystementwicklung. In Kapitel 3 werden die konzeptionellen Grundlagen für die ***Methode***

der semantischen Komposition dargestellt und erläutert. Kapitel 4 stellt ein Werkzeug vor, welches für Forschung und Lehre an der Technischen Universität Darmstadt entwickelt wurde: **TKMS - Terminologiebasiertes Komponenten-Management-System**. Mit diesem Werkzeug können Anwendungssystemkomponenten sprachbasiert verwaltet werden. Zudem ist es möglich mit den Mitteln der semantischen Komposition durch Variantenkonstruktion einfache browserbasierte betriebswirtschaftliche Anwendungen zu konfigurieren. Anhand eines Beispiels wird der Einsatz der ***Methode der semantischen Komposition*** ganzheitlich dargestellt und es wird damit ein Ausblick auf die zukünftige Entwicklung von betriebswirtschaftlichen Standard-Anwendungssystemen gegeben. Die Schlussbetrachtungen in Kapitel 5 fassen die wesentlichen Erkenntnisse der Arbeit für ein abschließendes Urteil zusammen.

Kapitel 2 Konstruktionslehre

Die Anwendungssystementwicklung besitzt ihre begrenzenden Faktoren. Lässt man technologische Grenzen außer Acht, so ergeben sich dennoch genügend Anforderungen, die direkt aus den verschiedenen Facetten der Aufgabenstellung für die Entwicklung von betriebswirtschaftlichen Anwendungssystemen ableitbar sind. Die Ziele der Anforderungen stehen teilweise zueinander im Konflikt. Unter Zielkonflikten sind hierbei Wechselwirkungen zwischen voneinander abhängigen Zielen zu verstehen, die bei einem positiven Annähern an ein Ziel negative Auswirkungen auf eines oder mehrere andere Ziele bzw. deren Erfüllung erzeugen.

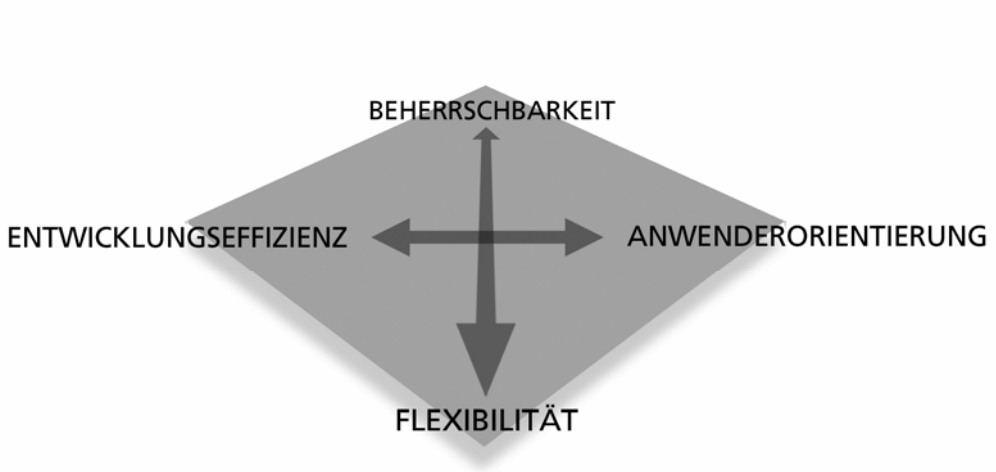


Abbildung 1: Magisches Viereck der Zielkonflikte in der Anwendungssystem-entwicklung

Beherrschbarkeit steht für die Wünsche des Herstellers nach überschaubarer Komplexität in der Programmerstellung, d.h. beherrschbare Prozesse zur Produktion und Qualitätssicherung. Ein Unternehmen, welches das Anwendungssystem einsetzt, versteht darunter einfache Betriebsbedingungen mit wenig Risiko und geringem Aufwand für Administrations-Ressourcen. **Anwenderorientierung** wird mit intuitiver Benutzbarkeit und dem Gefühl der Anwender, dass die Software genau so funktioniert, wie sie es wünschen, gleichgesetzt [Cooper99]. Zudem muss damit für die Anwender eine Transparenz bezüglich der integrativen Auswirkungen ihrer Handlungen und eine effiziente Arbeitsweise entlang der Wertschöpfungskette ihres Unternehmens gewährleistet werden. **Flexibilität** wiederum stellt sicher, dass ein Anwendungssystem möglichst viele unterschiedliche Verwendungsweisen unterstützen kann und unterschiedliche Lösungsmöglichkeiten für eine betriebliche Wertschöpfung zulässt. Sie führt im Herstellungsprozess zu Komplexität durch Entwicklung von Programmstrukturen, die den vielfältigen Verwendungsanforderungen gerecht werden. **Entwicklungseffizienz** steht für ein positives Verhältnis von Aufwand und Ertrag, bedingt durch marktgerechte Investitionsbeschränkung in der Herstellung von Anwendungssystemen. Gerade dieses Ziel verhindert häufig eine anwenderorientierte Lösung, denn die Wünsche der Anwender nach „einfacher Benutzbarkeit“ unter komplexen betriebswirtschaftlichen Rahmenbedingungen kann nur durch einen intensiven Einsatz von Entwicklungsressourcen zum Erstellen von komplexitätsverbergenden Lösungen erreicht werden.

Die Gegensätzlichkeit der Ziele ist begründet durch die verschiedenen Interessen der beteiligten Gruppen, die unabhängig voneinander ihre Anforderungen an ein Anwendungssystem stellen. Betriebswirtschaftliche Anwendungssysteme sind Wirtschaftsgüter, die unter bestimmten Voraussetzungen[1] auf dem Weltmarkt angeboten und verkauft werden können. Sowohl die Herstellung als auch der Einsatz von Anwendungssystemen ist durch vorwiegend wirtschaftliche Interessen begründet. Die Interessengruppen, ihre Anforderungen sowie die daraus abgeleiteten Ziele sind in der folgenden Tabelle dargestellt:

Interessengruppe	**Anforderungen**	**Ziele**
Management eines Softwareunternehmens	Kostengünstige Herstellung eines Produktes mit guter Marktakzeptanz und langfristigem Markterfolg	Beherrschbarkeit Anwenderorientierung Flexibilität Entwicklungseffizienz
Entwickler	Technologische Machbarkeit, Erweiterbarkeit, optimale Wartung	Beherrschbarkeit Flexibilität
Produktmarketing	Befriedigung der Kunden- bedürfnisse, gute Zielmarktkongruenz, gute Marktpreise	Anwenderorientierung Entwicklungseffizienz
Beratung und Vertrieb	Flexible Anpassung an Kundenbedürfnisse in Einführung und Betrieb	Flexibilität Anwenderorientierung
Kunde (Anwender)	Stabilität und Zuverlässig- keit, Erfüllung seiner aktuellen und zukünftigen Bedürfnisse (Sicherung seines langfristigen Geschäftserfolges), gutes Preis-Leistungsverhältnis für das Gesamtprodukt, bestehend aus Software, Hardware mit Infrastruktur und Beratung	Beherrschbarkeit Anwenderorientierung Flexibilität

Tabelle 1: Anforderungen an die Herstellung und den Einsatz von Anwendungssystemen

[1] Die Voraussetzungen sind beispielsweise Mehrsprachen-Fähigkeit, Fähigkeit zur Anpassung an länderspezifische Gesetze und geschäftsübliche Handlungsweisen, Mehrwährungsfähigkeit, länderspezifische Technologieunterstützung und Berücksichtigung von soziokulturellen Eigenheiten (Zeichensätze, Adressen, etc.).

Für Produkte mit derart differenzierten und gegensätzlichen Anforderungen von Seiten des Managements, der Herstellung, des Verkaufs und des Kunden muss die Konstruktion nach den Prinzipien der Ingenieurwissenschaften erfolgen, andernfalls ist ihre Entwicklung nicht mehr beherrschbar. Schon auf der ersten Software Engineering Konferenz in Garmisch 1968 wurde das Bedürfnis nach einer Konstruktion von Software, die einer technischen Konstruktion ähnlich ist, erkannt [McIlroy69]. Für eine Untersuchung der Möglichkeit technische Konstruktionsprinzipien auf die Anwendungssystementwicklung anzuwenden, werden im nächsten Abschnitt die wesentlichen Grundlagen für eine solche technische Konstruktionslehre zusammengefasst.

2.1 Technische Konstruktionslehre

In der technischen Konstruktionslehre verfügt man über eine langjährige Erfahrung in der Entwicklung und Konstruktion von Geräten, Maschinen, Anlagen und Systemen. Eine Konstruktionsmethode ist im Grunde genommen ein strukturierter Problemlösungsprozess für die Entwicklung von technischen Systemen. Verschiedene Ausführungen von Konstruktionsmethoden sind dadurch begründet, dass die Methodenentwickler einerseits aus unterschiedlichen Berufsfeldern kommen und dadurch unterschiedliche Erfahrungshintergründe besitzen [Pahl/Beitz93, 29]. Andererseits stellen sich verschiedene Aufgaben bezüglich der zu entwickelnden Produkte und ihrer Komplexität. Oder es werden die Faktoren zur Beeinflussung der Wertschöpfung unterschiedlich bewertet.

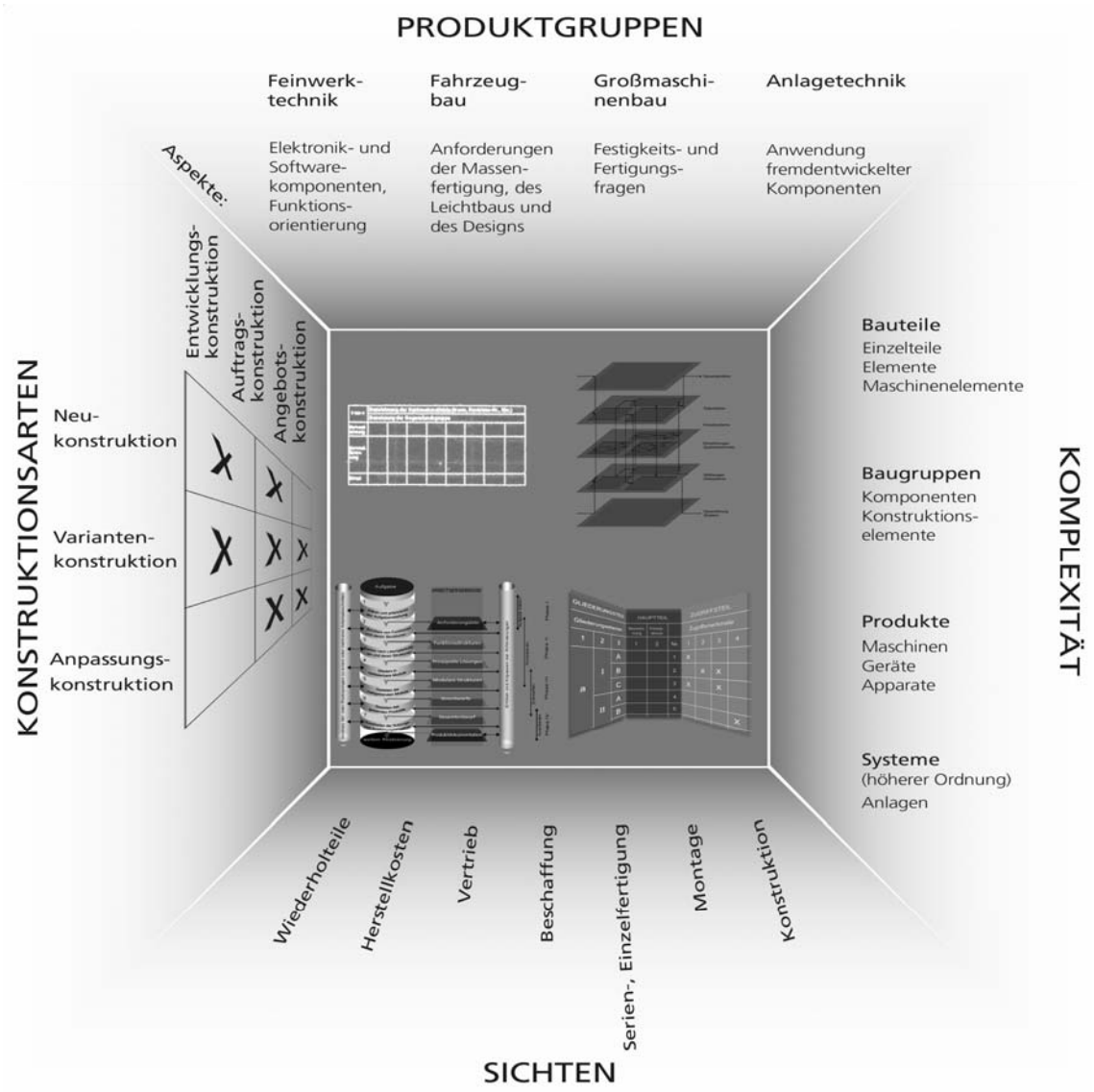


Abbildung 2: Übersicht der Einflussfaktoren auf die Konstruktionsmethodik (eigene Darstellung aufbauend auf [Pahl/Beitz93, 29, Steinhilper/Röper94, 7])

Konstruktionsobjekte der technischen Konstruktion sind immer technische Systeme. Aus der Systemtechnik kommend lassen sich viele Erkenntnisse der Systemanalyse und Systemsynthese auf die Konstruktionsmethodik übertragen [Steinhilper/Röper94, 1ff].

Definition:

> **Technisches System**, Gesamtheit von der Umgebung abgrenzbarer (Systemgrenze), geordneter und verknüpfter Elemente, die mit dieser durch technische Eingangs- und Ausgangsgrößen in Verbindung stehen [VDI 2221-93].

Es ist wichtig, ein System in seiner Ganzheit zu verstehen und gleichzeitig seine einzelnen Elemente für die Konstruktion und Herstellung differenziert betrachten zu können.

Methodisch gesehen ist der Vorgang des Konstruierens nur ein Optimieren unter vorgegebenen Zielen mit sich zum Teil widersprechenden Bedingungen und sich im Zeitverlauf ändernden Anforderungen. Die Tätigkeit eines Konstrukteurs wird als schöpferisch-geistige Arbeit verstanden, die ein sehr breit angelegtes Grundlagenwissen erfordert. Wissenschaftliche Vorbildung ist auf den Gebieten der Mathematik, Physik, Chemie, Mechanik, Wärme- und Strömungslehre, Elektrotechnik, Fertigungstechnik, Werkstoffkunde und Konstruktionslehre sowie der Betriebswirtschaft, Informatik und des jeweiligen Anwendungsbereichs erforderlich [Pahl/Beitz93, Breiing/Flemming93].

2.1.1 Methodisches Vorgehen

Sicher wird eine noch so gute Konstruktionsmethode die Fähigkeiten eines genialen oder auch nur durchschnittlichen Ingenieurs niemals voll ersetzen können, aber durch methodisches Vorgehen können beide in ihrer Effektivität erheblich unterstützt und angeregt werden. [...] Das Konstruieren würde, falls es je möglich wäre rein systematisch zu konstruieren, sehr viel von seiner Attraktivität verlieren.

RUDOLF KOLLER[2]

Die Vorgehensweise eines Ingenieurs zur Durchführung eines Konstruktionsprojektes ist nie vollständig standardisierbar. Zu viele Einflussfaktoren und Restriktionen (Herstellkosten, technische Machbarkeit, unbekannte oder sich laufend verändernde Anforderungen etc.) erfordern ein flexibles, der Situation angepasstes Vorgehen. Dennoch ist es gerade die Stärke der technischen Konstruktionsbereiche, dass sich sowohl für die Lehre und Ausbildung von jungen Ingenieuren als auch für die tägliche Konstruktionspraxis ein generelles Vorgehensmodell entwickeln und normieren ließ

[2] Zitiert nach [Ortner97b, 14]

(siehe Abbildung 3). Aus diesem Vorgehen abgeleitet haben sich verschiedene Konstruktionsablaufpläne mit detaillierten Tätigkeitsangaben und Ergebnisdokumenten herausgebildet. Bei Roth [Roth94 Bd.1, 42f] wird eine Gegenüberstellung der wesentlichen Ablaufpläne von Pahl und Beitz [Pahl/Beitz93, 80ff], Koller [Koller85, 19ff] und der VDI Richtlinie 2221 [VDI 2221-93] aufgeführt und diskutiert. Eine weitere Variante findet sich in Breiing und Flemming [Breiing/Flemming93]. Hier wird der Konstruktionsprozess eingebettet in einen entwicklungstechnischen Gesamtprozess für ein Produkt in einer Grafik dargestellt. Im generellen Vorgehensmodell der VDI-Richtlinie 2221 wird eine Einbettung des Konstruktionsprozesses in den Produktkreislauf separat dargestellt.

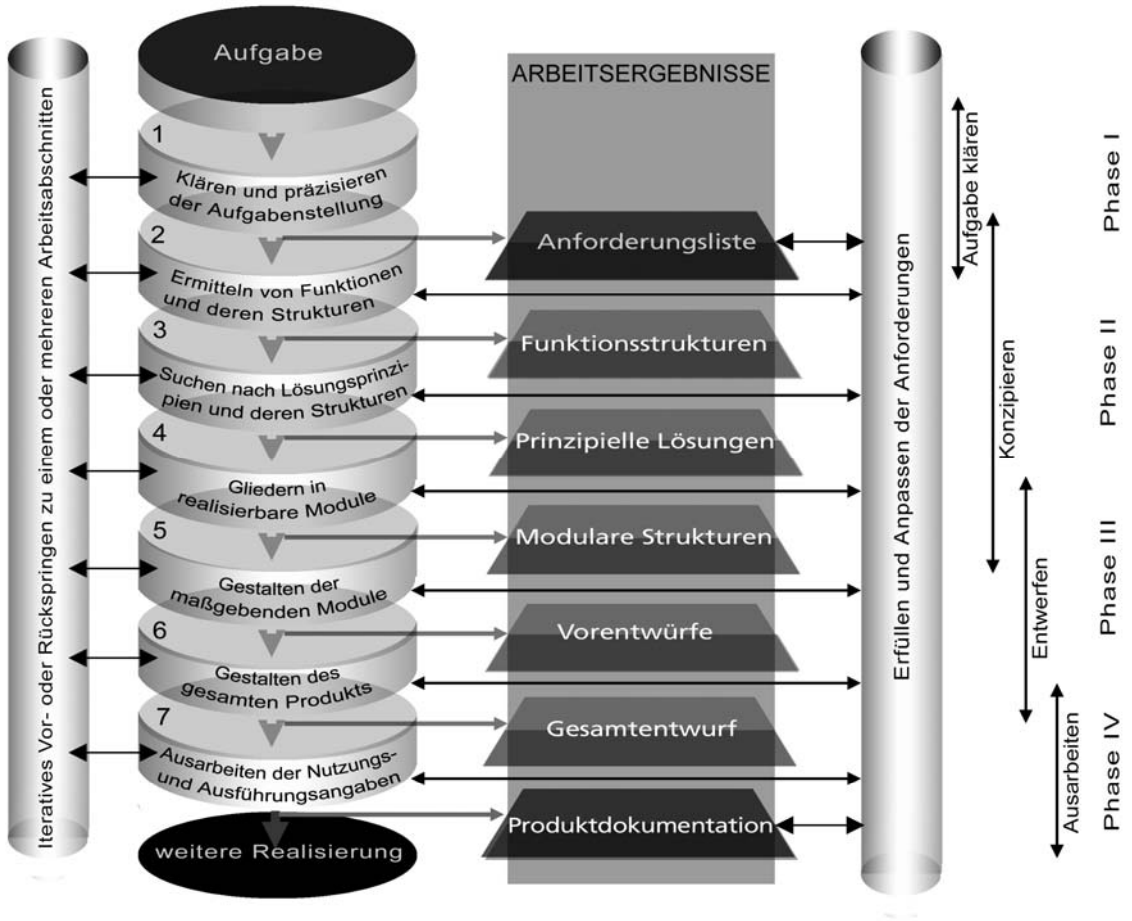


Abbildung 3: Generelles Vorgehensmodell für die technische Entwicklung und Konstruktion [vgl. VDI 2221-93]

In dem o.g. Vorgehensmodell werden sieben Arbeitsschritte unterschieden. Die Bezeichnung und Abgrenzung der einzelnen Konstruktionsphasen wird offen gehalten

und kann je nach Produkt und Problemstellung variiert werden. Für die Konstruktion mit Katalogen ist in der VDI-Richtlinie [VDI 2222-82] eine Essenz aus Roths und Kollers Ablaufplan umgesetzt und definiert die Phasen „Planen“, „Konzipieren“, „Entwerfen“ und „Ausarbeiten“, die im Wesentlichen in allen Ablaufplänen wiederzufinden sind[3]. Die Planungsphase beinhaltet ein Auswählen der Aufgabe bedingt durch Trendstudien, Marktanalysen, Forschungsergebnisse, Kundenanfragen, Vorentwicklungen, Patente oder Umweltschutzaspekte. In VDI [VDI 2222-82] sind die einzelnen Tätigkeiten mit ihren Entscheidungspunkten ausführlich dargestellt. Entscheidungspunkte sind wichtig für ein methodisches Konstruieren, denn sie sind maßgeblich für das iterative Vor- und Zurückspringen im Ablauf der Konstruktion. Erwähnenswert ist, dass allen Vorgehensweisen keine ausschließlich sequentielle Bearbeitung zugrunde liegt.

Zu den Arbeitsschritten 1, 2 und 4 sollten an dieser Stelle noch einige Aussagen hervorgehoben werden. Bemerkenswert ist im Schritt 1 Kollers Trennung von Zweckbeschreibung und Bedingungen (Restriktionen) der Aufgabenstellung [Koller85, 12ff]. Eine Zweckbeschreibung ist die Beschreibung der Fähigkeiten eines zu entwickelnden Produktes bzw. Systems, ohne dabei die Lösungsmöglichkeiten einzuschränken. Daneben werden Bedingungen, die eine Restriktion der Lösungsmenge bedeuten, separat aufgeführt. Diese Bedingungen resultieren daraus, dass jedes technische System in der Regel wiederum Bestandteil eines umfassenderen Systems ist (Schnittstellen) und für einen bestimmten Markt entwickelt wird (Marktbedingungen) oder gesetzlichen Vorschriften genügen muss (Gesetze, Normen). Auch der Schritt 2 wird allgemein in zwei Tätigkeiten unterteilt: die Ermittlung der Gesamtfunktion (Gesamtaufgabe) und anschließend die Suche nach geeigneten Teilfunktionen (Teilaufgaben), die weitere Konkretionen der Lösungen ermöglichen. Eng damit verknüpft ist die Tätigkeit in Arbeitsschritt 4. Dort werden die modularen Strukturen eines Systems festgelegt. Dafür sind die Teilaufgaben ein wesentlicher Gestaltungsfaktor.

In Roth [Roth94 Bd.1, 34ff] werden alle sieben Arbeitsschritte in der Form des o. a. Vorgehensmodells detailliert aufgeführt.

---

[3] Die Phase „Aufgabe klären“ ist als Zwischenschritt zwischen Planung und Konzeption anzusehen. Falls keine Planung vorangestellt ist, so wird die Aufgabenklärung als eigenständige Phase definiert.

2.1.2 Problemlösung

Konstruktionen für technische Systeme sind gekennzeichnet durch vielfältige Probleme und Aufgaben. Dörner [Dörner87, 10] gliedert Probleme in drei Komponenten:

1. Unerwünschter Anfangszustand $s\alpha$
2. Erwünschter Endzustand $s\omega$
3. Barriere, die die Transformation von $s\alpha$ in $s\omega$ im Moment verhindert.

Aufgaben sind gegen Probleme abzugrenzen. Sie stellen geistige Anforderungen dar, für deren Lösung die Methoden grundsätzlich bekannt sind [Dörner87, 19]. Bei Aufgaben fehlt also die dritte Komponente, nämlich die Barriere. Bezüglich der Abgrenzung von Problem und Aufgabe stellen Pahl und Beitz [Pahl/Beitz93, 58] fest, dass sich bei manchen Konstruktionsaufgaben, werden sie in Teilaufgaben gegliedert, schwierige Teilprobleme ergeben. Umgekehrt kann ein Problem durch eine neuartige Kombination von Teilaufgaben (und ihren Teillösungen) gelöst werden. Die Barrieren der Problemlösung sind unterschiedlicher Art. So können beispielsweise die Mittel zur Transformation unbekannt sein (Syntheseproblem), oder es gibt eine sehr große Zahl von Lösungen (Kombinations- oder Auswahlproblem). Eine weitere Möglichkeit besteht darin, dass die Komponente „erwünschter Endzustand“ nicht genau definiert werden kann.

Dörner unterscheidet weitere Kriterien, die Einfluss auf die Entwicklung von Systemen[4], je nach ihrem Einsatzbereich, nehmen können [Dörner87, 18ff]:

- Komplexität
- Dynamik
- Vernetztheit
- Transparenz
- Grad des Vorhandenseins freier Komponenten

[4] Dörner spricht eigentlich von Sachverhalten oder Situationen. Für technische Bereiche ist eine synonyme Verwendung von Systemen jedoch zulässig.

Komplexität definiert Dörner als die Anzahl von Komponenten, die in einem System unterschieden werden können und die Vielfalt der Verknüpfungen zwischen den Komponenten.[5]

Unter **dynamischen** Situationen versteht Dörner, sich permanent verändernde Bedingungen. Sie fordern von einem Problemlöser, dass er unter Zeitdruck handeln kann und in der Lage ist, Entwicklungen abzuschätzen.

Ein hoher **Vernetzungsgrad** wird erreicht, wenn die Variablen oder Merkmale eines Systems in hohem Maße voneinander abhängen und folglich keine isolierte Behandlung bzw. Beeinflussung möglich ist. Problemlöser müssen Nebenwirkungs-analysen durchführen, um ein Systemverhalten beherrschen zu können.

Die **Transparenz** eines Systems ist bestimmt durch die Möglichkeiten, einzelne Eigenschaften des Systems direkt zu erkennen und ihre Ausprägungen festzustellen. Existieren dabei Schwierigkeiten, so muss auf „Symptome“ zurückgegriffen werden, d.h. auf sichtbare Merkmale, die Rückschlüsse auf latente Merkmale ermöglichen. Ein Zusammenhang zwischen Symptomen und latenten Merkmalen ist jedoch nie ganz sicher zu bestimmen.

Freie Komponenten ermöglichen durch Kombination und Austausch von Komponenten eine effiziente Konstruktion und Wartung von Baugruppen. Sind wenig freie Komponenten vorhanden, so müssen existierende Komponenten an veränderte Verhältnisse angepasst werden, wodurch eine tiefere Einsicht in den Aufbau der Komponenten notwendig wird. Dadurch wird die Problemlösung erschwert.

Übersteigt die Komplexität ein elementares Niveau, so reichen häufig die intellektuellen Fähigkeiten eines Problemlösers nicht mehr aus.

---

[5] Bei Klaus wird **Komplexität** definiert als „Eigenschaft von Systemen, die durch die Art und Zahl der zwischen den Elementen bestehenden Relationen festgelegt ist; im Unterschied zur **Kompliziertheit** eines Systems, die sich auf die Zahl unterschiedlicher Elemente bezieht (Grad der Unterschiedlichkeit)“ [Klaus71].

Es müssen Maßnahmen zur Komplexitätsreduzierung und weitere „problemadäquate Vorgehensweisen“ ergriffen werden [Müller90, 12]:

- Modularisierung
- Aufsteigendes Vorgehen
- Adaptives Vorgehen
- Rückgriff auf Vorhandenes

2.1.2.1 Modularisierung

Die Modularisierung ist in der Kybernetik als „wirksame und wirtschaftliche Vorgehensweise zur Lösung komplexer Probleme“ erkannt worden und wird dabei als „Aufgliederung des Problemlösungsprozesses in parallel laufende Lösungswege“ umgesetzt [VDI 2221-93].

In der technischen Konstruktion wird durch die Konstruktionshandlung der Partition eine Gesamtaufgabe in Teilaufgaben bzw. ein Gesamtproblem in Teilprobleme zerlegt. Die Zerlegung schreitet fort bis elementare Aufgaben (Einzelprobleme) gefunden sind. Für die elementaren Aufgaben können wesentlich einfacher Lösungen entwickelt bzw. ein Lösungsspektrum aus alternativen Lösungen systematisch erarbeitet werden. Auch das Verwenden von vorhandenen, bewährten Teillösungen wird besser unterstützt.

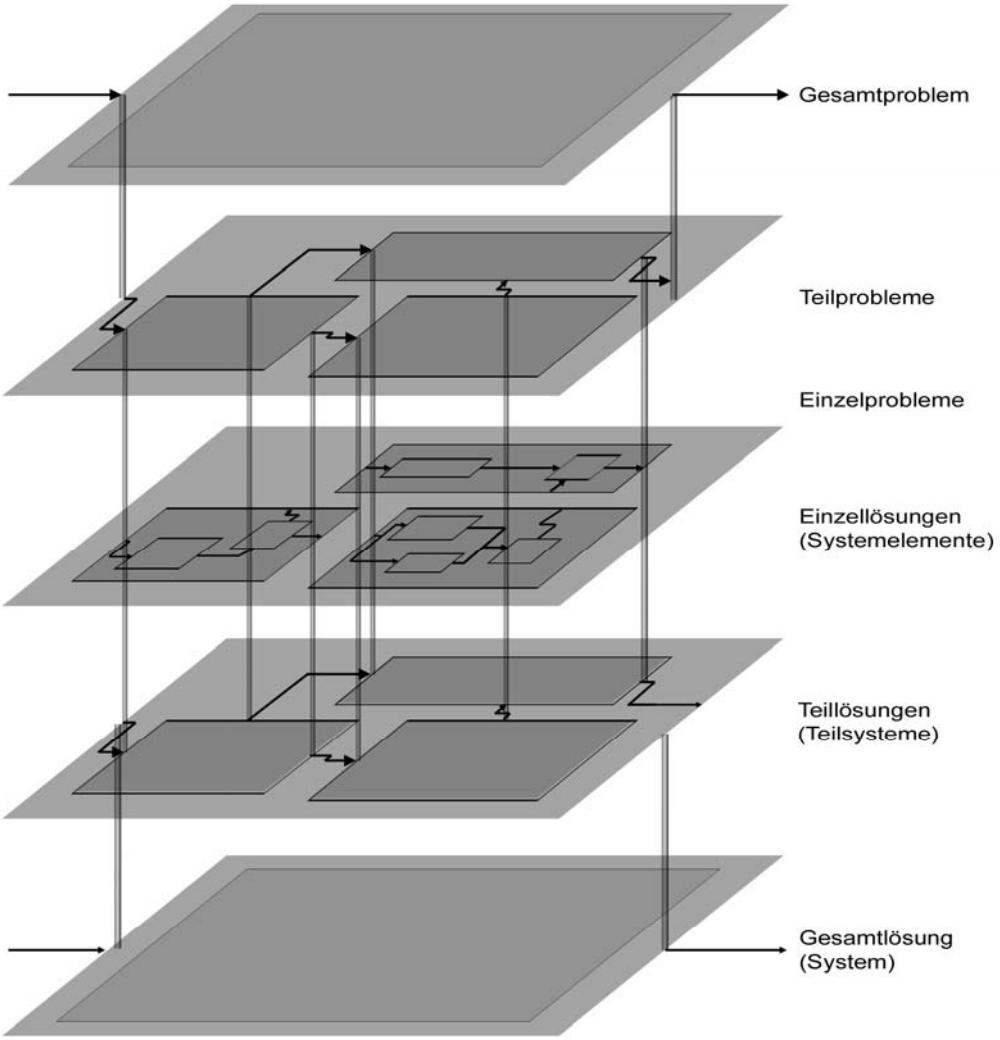


Abbildung 4: Problempartition und Lösungskomposition [VDI 2221-93]

Sind die Lösungen gefunden, so werden sie durch Komposition wieder schrittweise zu einer Gesamtlösung zusammengeführt. Das Gesamtsystem darf nicht aus den Augen verloren werden. Problematisch ist dabei die eingeschränkte Kombinationsfähigkeit (Unverträglichkeit) von Lösungen. Eine systematische Nutzung der Modularisierung wird in Abschnitt 2.1.4 gezeigt.

2.1.2.2 Aufsteigendes Vorgehen

Bei umfangreichen Konstruktionsaufgaben ist es nicht möglich, alle Anforderungen, Probleme, Teilaufgaben und Lösungen auf einmal zu bearbeiten. Ein schrittweises Vorgehen zur Erfassung und Verarbeitung der Informationen rund um eine Konstruktion - ausgehend vom Kern einer Aufgabe - hat sich bewährt. Die von der „Problemtiefe“ aufsteigenden Schritte hin zur „Erkenntnishöhe“ können durch folgende Anweisungen koordiniert werden:

1. vom Abstrakten zum Konkreten
2. vom Teil zum Ganzen
3. vom Wesentlichen zum weniger Wesentlichen bzw. von den Hauptproblemen zu den Teilproblemen oder vom Zentrum zur Peripherie [Müller90, 12, VDI 2221-93]

Die ersten beiden Anweisungen entsprechen den grundlegenden Konstruktionshandlungen *„Abstraktion“* (Konverse: *„Konkretion“*) und *„Komposition“* (Konverse: *„Partition“*) [VDI 2221-93, Pahl/Beitz93, Müller90, Wedekind/Ortner80]. Unter 1. wurde nur die *„Konkretion“* angesprochen, denn ein Erarbeiten der Problemstellung wird hier nicht berücksichtigt. Betrachtet man einen Konstruktionsprozess umfangreicher, nicht nur die Problemlösung, sondern auch die Aufgabenstellung oder das Aufspannen eines Lösungsspektrums, so stellt man fest, dass die *„Abstraktion“* und anschließend die *„Konkretion“* in Wechselwirkung mit *„Komposition“* und *„Partition“* erfolgt.

Die Abstraktion geht von den Anforderungen aus und eliminiert das Unwesentliche bis die Kernforderungen erkannt sind. Danach folgt eine Zerlegung in Teilaufgaben (Partition). Diese werden dann realisiert (Konkretion) und anschließend wieder zu einem Ganzen zusammengeführt (Komposition) [Roth94 Bd. 1]. Die Zusammenführung der Teillösungen zu einer Gesamtlösung ist ein notwendiges Kriterium für die Zielerreichung der Konstruktionshandlung. Auf dieser Zusammenführung muss das Hauptaugenmerk liegen.

Eine Abstraktion geht immer schrittweise vor sich. Abhängig vom Anwendungsgebiet wird nach Gemeinsamkeiten und Veränderlichkeiten unterschieden. Gemeinsamkeiten bilden dabei die übergeordneten Abstraktionsebenen (Generalisierung). Wird die Form

der Beschreibung von Gegenständen und Ideen als „intensionale Begriffsbestimmung“ (Merkmalbeschreibung) betrachtet, so erfolgt eine Eigenschaftsreduzierung der Begriffe in Richtung zunehmender Abstraktion. Dies bedeutet, dass der Detaillierungsgrad der Begriffe abnimmt.

Dieses Prinzip der Abstraktionsebenen kann auch auf die Kompositionsrelationen von Gegenständen übertragen werden. Alle Gegenstände der realen Welt können in ihre Bestandteile zerlegt werden. Ihre Betrachtung als Ganzheit bzw. Baugruppe ist bedingt durch die Denkstruktur des Menschen. Nach Mayer [Mayer79] ist der Mensch geneigt, alles zu Denkende und zu Betrachtende in Mustern bzw. in attributbasierten Konzepten, eingeordnet in Kategorien, zu erfassen.

Die Ebenenbildung einer schrittweisen Komposition dient der Verminderung von Komplexität. In diesem Sinne müssten die Ebenen eigentlich Komplexionsebenen genannt werden [Müller90]. Dörner [Dörner87] bezeichnet die verschiedenen Ebenen der Komposition als Auflösungsgrade. Damit wird die Sichtweise in Richtung Partition ausgedrückt. Die umfangreichere Information „aus welchen Teilen besteht eine Baugruppe“ stellt somit einen höheren Auflösungsgrad dar.

In der folgenden Darstellung wird der hierarchische Aufbau von Gegenständen bzw. von optionalen Gegenständen in einer Kompositionshierarchie gezeigt, wobei die Ebene der nächstübergeordneten Baugruppe in der Entwurfsphase als nächsthöhere Kompositionsebene bezeichnet wird.

Kompositionshierarchie und Abstraktionshierarchie unterliegen völlig anderen Beziehungen zwischen den Ebenen, was durch die englischen Beziehungsbezeichnungen aus dem Bereich der Wissensrepräsentation gut ausgedrückt ist [Reimer91,79ff]:

- „is-a“ für die Abstraktionsrelation und
- „has-part“ für die Kompositionsrelation.

Abstraktion und Komposition, basierend auf „Gleichheit“ und „Abhängigkeit“ als Grundbeziehungen, lassen sich jedoch nicht nur auf diese zwei Relationstypen beschränken [Ortner83, Ortner95, Schienmann97]. Abstraktion und Komposition werden unter dem speziellen Blickwinkel der Anwendungssystementwicklung in Abschnitt 3.3.4 ausführlich behandelt.

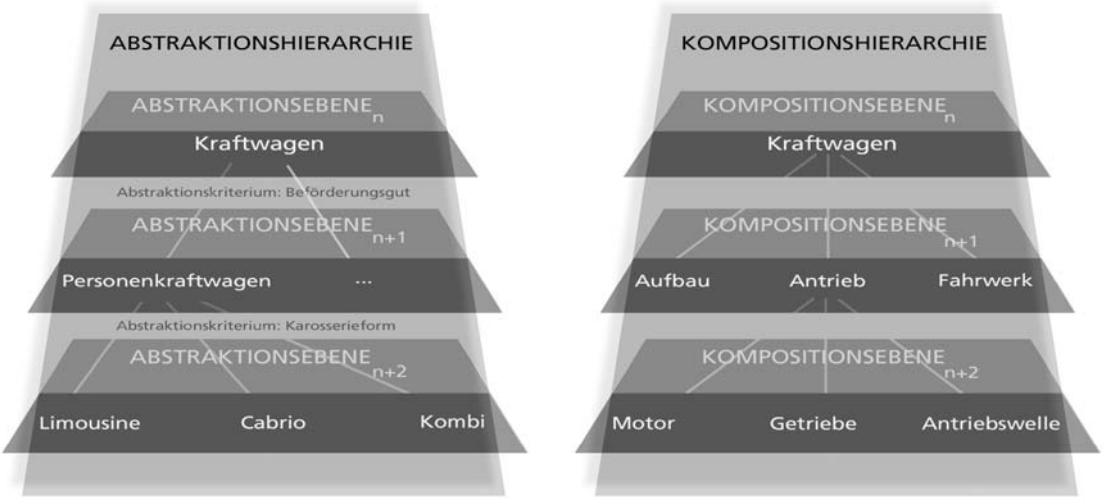


Abbildung 5: Strukturelle Darstellung von Abstraktion/Konkretion und Komposition/ Partition [6]

Eine strukturelle Darstellung der Kompositionsrelationen kann man als Ergebnis eines bestimmten Konstruktionsprojektes sehen. Dann entspricht sie dem Systemstammbaum der fertigen Konstruktion [Breiing/Flemming93]. Oder sie wird für die Repräsentation von optionalen Kompositionen einer Bauteilesammlung verwendet. Dies entspricht dann beispielsweise der polyhierarchischen Gliederung eines Zugriffssystems auf ein Baukastensystem (siehe Abschnitt 3.4.3). Optionale Kompositionen sind Referenzmodelle, die dem Konstrukteur alternative Komponentenkonfigurationen anbieten.

Die Zielsetzungen der Konstruktionshandlung *„Partition“* können sehr unterschiedlich sein. Es können die Bedürfnisse der Produktion, der Montage, der Lagerung, der Beschaffung, der Ersatzteilherstellung oder der Konstruktion im Vordergrund stehen [Scheer94, Richter84, Wedekind/Müller81]. Hat man sich auf die Zerlegung mit Blick auf eine Variantenkonstruktion mittels Baukastensystemen fokussiert (siehe Abschnitte 2.1.4 und 2.1.6), so ist die Partition ein Suchen nach Gemeinsamkeiten, welche mit variierenden Komponenten kombiniert werden können. Schrittweise werden dabei

[6] Die Ebenenbildung beginnt bei Hierarchien oben. Die erste Ebene ist also die oberste Ebene.

Komponenten in ihre Elemente zerlegt (Kompositionsebenenwechsel), um variabel auf Anforderungen reagieren zu können.

2.1.2.3 Adaptives Vorgehen

Nach Müller [Müller90] ist der intellektuelle Fortschritt (Aufstieg) bei der Systementwicklung nur durch geeignete Wechsel von Abstraktion und Komposition zu erreichen. Keine sequentielle und auch keine zyklische bzw. spiralförmige Arbeitsweise ist alleine ausreichend. Müller beobachtete bei erfolgreichen Entwicklungsingenieuren eine ausgeprägte Parallelarbeit in verschiedenen Abstraktionsebenen und zu mehreren Teilproblemen, zwischen denen die Ingenieure sich oszillierend bewegen. Einem ersten ganzheitlichen Erfassen der Aufgabenstellung folgen wechselnde Abstraktionsbildung (jedoch nur soweit sinnvoll!) und Konkretion sowie Partition und Komposition, bis die fertige Systemlösung entsteht.

2.1.2.4 Rückgriff auf Vorhandenes

Müller [Müller90, 12] stellt maximalen Einsatz und Nutzung vorhandener, bewährter Lösungen, ohne dabei die Aufgabenstellung unzulässig zu reduzieren, als ein notwendiges Kriterium für die Effektivität eines Ingenieurs heraus.

In technischen Konstruktionsbereichen stehen für viele Arbeitsschritte und Tätigkeiten der Konstruktion externe Wissensspeicher zur Verfügung. Dies können einfache Hilfsmittel wie Formulare mit Beispielen, Begriffslisten oder Suchmatrizen für die Unterstützung der Klärung der Aufgabenstellung [Roth94 Bd. 1, 57ff] sein. Darüber hinaus werden manuelle oder elektronische Konstruktionskataloge in allen Phasen der Konstruktion eingesetzt [Roth94, Bd. 1 u. 2, VDI 2222-97, VDI 2222-82]. Diese Kataloge sind entweder produktneutrale, methodisch erstellte Normkataloge oder sie werden von Produktlieferanten in Form von Produktkatalogen bereitgestellt.[7] Die nächste Stufe elektronischer Unterstützung sind Expertensysteme für die Entwicklung und Konstruktion [VDI-Berichte93].

---

[7] Beispiel: Norelem: Flexibles Normteilesystem für die Konstruktion von Fräsvorrichtungen, Bohrvorrichtungen und Spannsystemen. Der Katalog der Firma Norelem aus Markgröningen unterstützt die modulare Konstruktion von Vorrichtungen (Betriebsmittel) [Norelem96].

Externe Wissensspeicher unterstützen meist keine durchgehende Abstraktion, sondern nur Abstraktionen innerhalb von bestimmten Abstraktionsstufen (z.B. Anforderung, Funktion, Prinzip, Gestalt). Innerhalb dieser Stufen wird dann weiter spezialisiert, bis die geeignete Lösung gefunden wird. Mit dieser Lösung werden die weiteren Konstruktionshandlungen durchgeführt [Grabowski et.al. 93].

A) Kataloge

Ordnungsschemata helfen, einen Lösungsbereich zu strukturieren und unterstützen dadurch seine vollständige Abbildung und das Entwickeln von neuen Lösungsmöglichkeiten. Die Suche nach vorhandenen Lösungen wird durch ein Ordnungsschema ebenfalls erleichtert [Pahl/Beitz93, 102ff, Dreibholz75].

Der strukturelle Aufbau von Konstruktionskatalogen wurde in den VDI-Richtlinien [VDI 2222-82] unter der Leitung von Prof. Dr.-Ing. K. Roth normiert sowie in Roth [Roth94 Bd. 2] in aller Ausführlichkeit dargestellt und wird im Folgenden auf dieser Grundlage erläutert.

Definition:

> **„Konstruktionskataloge** sind Informationsspeicher, die hinsichtlich ihrer Inhalte, ihrer Zugriffsmöglichkeiten und ihres Aufbaus auf das methodische Konstruieren zugeschnitten sind. Ihre besonderen Kennzeichen sind weitgehende Vollständigkeit, klare Gliederung (Systematik) und Existenz von Zugriffsmerkmalen“ [VDI 2222-82, 4].

Kataloge lassen sich inhaltlich in Bezug auf den Anwendungsbereich unterscheiden. Sie sind jedoch auch von ihrer Art her verschieden. Die drei wichtigsten Katalogarten sind:

- Objektkataloge
- Operationskataloge
- Lösungskataloge

Objektkataloge enthalten aufgabenunabhängige Elemente. Sie speichern Wissen, welches allerdings nicht spezifisch ganz bestimmten Funktionen (Aufgaben) oder Produkten zugeordnet ist. Der Ordnungsgesichtspunkt ist eine „Klasse von Mitteln“.

Objektkataloge können in strengem Sinne vollständig sein, beispielsweise können alle einstufigen Zahnradgetriebe als Elemente enthalten sein.

Operationskataloge beinhalten Operationen (Verfahrensschritte) und Operationsfolgen (Verfahren) für die Unterstützung des Konstruktionsprozesses sowie deren Anwendungsbedingungen und Einsatzkriterien. Beispielsweise sind dies Regeln zur Erzeugung von Komponenten-Varianten oder Verfahren zur Lösungsauswahl.

Lösungskataloge sind Konstruktionskataloge, die eine Zuordnung von Funktionen bzw. Aufgaben zu Lösungen ermöglichen. Diese Lösungen sind Komponenten. Der Ordnungsgesichtspunkt ist eine „Aufgabe“ oder eine „Klasse von Aufgaben“. Lösungskataloge sollten möglichst umfassende Lösungssammlungen beinhalten. Eine der wichtigsten Aussagen über das Konstruieren mit Lösungskatalogen ist, dass sich bei der Konstruktion mit fertigen Lösungen, dies entspricht einer Konstruktion durch Konfiguration von Komponenten, Wegstrecken des ursprünglichen Konstruktionsprozesses überbrücken lassen. Durch die Wiederverwendung von bereits konstruierten Lösungen kann der Konstruktionsprozess für bestimmte Teilaufgaben auf die Auswahl der geeigneten Lösungen aus dem Katalog verkürzt werden.

Wichtig ist noch ein Hinweis bezüglich der Gliederungsmöglichkeit des Konstruktionsprozesses nach Konstruktionsaufgaben. Dieses Gliederungskriterium ist nur für Lösungskataloge sinnvoll, denn nur bei diesen besteht ein enger Zusammenhang zwischen Kataloginhalt und bestimmten Konstruktionsaufgaben. Je Aufgabe sind im Allgemeinen unterschiedliche Merkmale für die Lösungsauswahl bestimmend. Roth hat gegenüber der VDI-Richtlinie noch eine weitere Katalogart eingeführt - den Beziehungskatalog. Darunter ist die Verwaltung von aus mindestens zwei Objekten bestehenden Lösungen zu verstehen. Diese Katalogart verwaltet aus elementaren Lösungen bestehende Baugruppen (konfigurierte komplexe Lösungen).

B) Aufbau von Katalogen

Konstruktionskataloge bestehen im Wesentlichen aus drei Elementen:

- Gliederung
- Hauptteil
- Zugriffsmerkmale

Roth sieht die *Gliederung* als Instrument für eine widerspruchsfreie Unterteilung des Kataloginhaltes (Hauptteil) mit dem Ziel der Vollständigkeit. Sie entspricht einer Abstraktionshierarchie. Der *Hauptteil* entspricht dem Kataloginhalt und enthält je nach Katalogart Objekte, Operationen oder Lösungen. Eine Anforderung von Roth ist es, den in Form von Skizzen, Gleichungen oder Texten dargestellten Inhalt, jeweils auf derselben Abstraktionsebene zu präsentieren.

Ein Zugriff auf den Kataloginhalt mittels Merkmalen erfolgt im *Zugriffsteil*. Zugriffs- oder beschreibende Merkmale sind charakterisierende Eigenschaften, die anwendungs- oder branchenabhängig den Katalogelementen zugeordnet werden, damit bei entsprechender Suche über ein so genanntes „Aussieben“ die günstigsten Lösungen gefunden werden können.

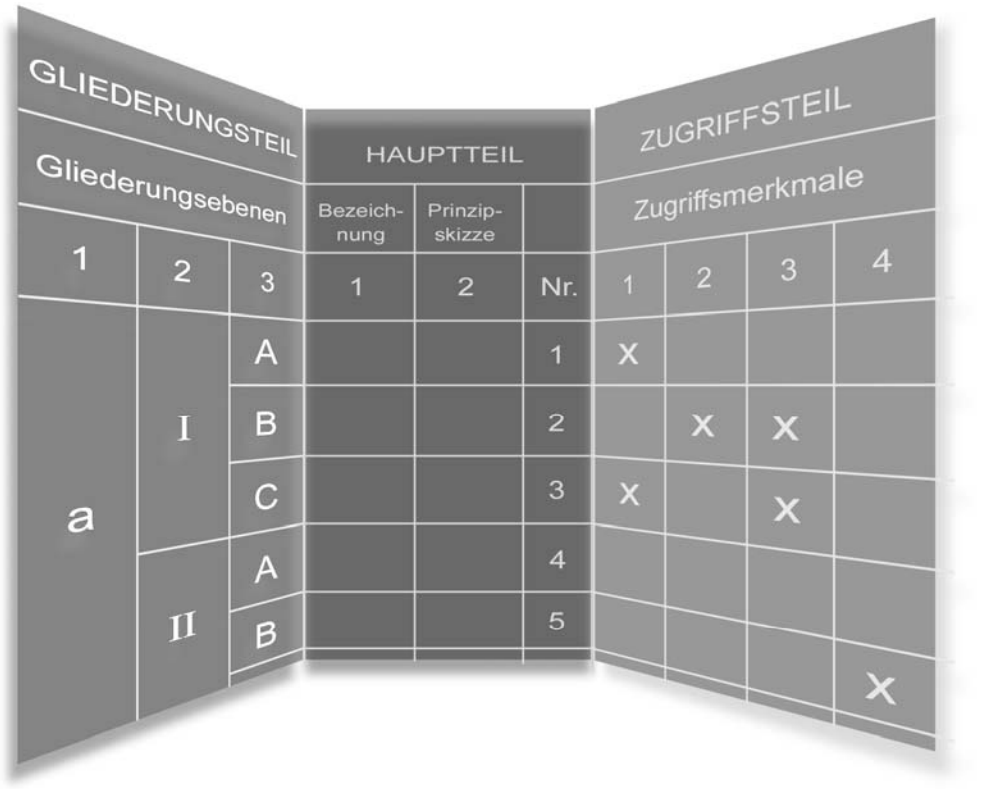


Abbildung 6: Aufbau eines Konstruktionskataloges mit "eindimensionalem" Gliederungsteil [vgl. Roth94 Bd. 2, 8]

Die Festlegung der Zugriffsmerkmale ist von den Gliederungsbereichen abhängig. Zugriffsmerkmale sollten die von den Gliederungsgesichtspunkten nicht erfassten Eigenschaften ansprechen. Sind die Anwendungsbereiche für Kataloge sehr groß, so

werden mit Übersichts- und Detailkatalogen ordnende Strukturen geschaffen. Die Gliederung eines Konstruktionskataloges lässt sich aus vorhandenen Systemen ableiten, die für einen Aufgabenbereich ein möglichst breites Spektrum an verschiedenen Lösungen beinhalten. Die Lösungen werden in eine Hierarchie eingeordnet, die entsprechend geeigneter Abstraktionskriterien aufgebaut ist. Anschließend wird versucht, zu den Gliederungsbereichen weitere Lösungen zu finden. Eine ausführliche Literaturrecherche in diesem Anwendungsbereich ist dabei notwendig, damit eine möglichst vollständige Abdeckung einer begrenzten Anwendungsdomäne erreicht werden kann. Der Aufbau von Katalogen sollte an einer Konstruktionsmethodik ausgerichtet, widerspruchsfrei und in Bezug auf die Anwendungsbereiche abgrenzbar sein. Geeignete Darstellungsformen der Lösungen im Hauptteil sind auszuwählen. Anschließend werden Zugriffsmerkmale für die auswahlunterstützenden Beschreibungen der Lösungen festgelegt. Wie eine strukturierte Übersicht von Gliederungs- und Zugriffsmerkmalen aussehen kann, ist für Teilgebiete der Physik in den VDI-Richtlinien und bei Roth ausführlich dargestellt. Der Aufbau von Konstruktionskatalogen soll anhand eines Beispiels weiter verdeutlicht werden.

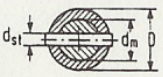
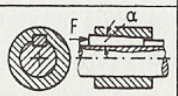
Gliederungsteil		Hauptteil				Zugriffsteil							Anhang
Art des Flächenschlusses	Art der Kraftübertragung	Gleichung	Benennung	Anordnungsbeispiel		Übertragbares Moment	Momentübertragung abhängig von	Aufnahme von Axialkräften	Wirkung bei Überlastung	Verbindung zentrierbar	Nabe axial verschiebbar	Nabe versetzbar	Anmerkungen
1	2	1	2	3	Nr	1	2	3	4	5	6	7	8
Normal (Formschluß)	Unmittelbar	$M_t = \frac{d_m}{2} A_{\tau ges} \tau_{zul}$ $M_t = \frac{d_m}{2} A_p p_{zul}$ M_t übertragbares Moment d_m mittlerer wirksamer Durchmesser	Profilwelle		1	groß	Formfaktor	nein	Bruch	ja	möglich	in Stufen möglich	—
	Mittelbar		Formelementverbindung		2	klein		möglich				möglich	einfache Montage
Tangential (Reibschluß)	Unmittelbar	$M_t = M_r = F_r \, d_m/2 = F_n \, \mu \, d_m/2$ α Keilwinkel M_r Reibmoment F_r Reibkraft d_{st} Stiftdurchmesser	Klemmsitz		3	klein bis groß	Temperatur, Rotationskräften, Axialkräften	ja	Rutschen	ja	nur bei $F_A > F_r$	stufenlos	—
	Mittelbar		Spannelement		4	mittel				möglich			Herstell- und Montageaufwand klein
Tangential und normal	Mittelbar	—	Vorgespannte Verbindung		5	klein		möglich	Bruch	nein	nein	möglich	—

Abbildung 7: Beispiel eines Konstruktionskataloges: Welle-Nabe-Verbindungen, Übersichtskatalog [Roth94 Bd. 2, 165]

2.1.2.5 Zusammenfassung Problemlösung

Komplexität kann man nicht wirklich verringern [Ashby74], sondern nur durch Abstraktion und Komposition verbergen bzw. beherrschen. Werden die Lösungskomponenten und Ihre Baugruppen entwickelt, so sollte auf eine Minimierung der Vernetzung geachtet werden, damit die Kombinierbarkeit und Austauschbarkeit von Lösungen erhalten bleibt. Das Ziel, die Transparenz zu erhöhen, sollte genauso verfolgt werden, wie die Schaffung freier Komponenten. Des Weiteren ist die Wiederverwendung von bewährten Lösungsideen und Lösungsprodukten eine Voraussetzung für methodisches und effizientes Konstruieren.

2.1.3 Systematische Lösungsentwicklung

Für die „systematische Lösungssuche“ hat Zwicky [Zwicky71] eine allgemeingültige Methode der Morphologie entwickelt. Sie wurde zu einer „interdisziplinären Methodenwissenschaft (Methodologie) für die Gestaltung des kreativen Denkens in geordneter Form” [Holliger82, 7] ausgebaut und ist heute die Grundlage für viele Methoden zum Finden neuer Lösungsvarianten und zum Darstellen von Lösungsspektren. Zwicky stellt in [Zwicky71, 88] einen *„Morphologischen Kasten“* vor, mit dessen Hilfe eine übersichtliche und vollständige Darstellung von Lösungsalternativen für ein gegebenes Problem möglich ist, sofern die beeinflussenden Parameter ausreichend gut bestimmt wurden. Zwickys Morphologischer Kasten ist ein vieldimensionales Schema für die Lösungsdarstellung. In der Regel werden jedoch zweidimensionale morphologische Matrizen verwendet [Breiing/Flemming93, 54]. Wobei mehrdimensionale Schemata auf zweidimensionale zurückgeführt werden können [Dreibholz75]. Zwicky beschreibt den Aufbau eines Morphologischen Kastens folgendermaßen [vgl. Zwicky71, 88]:

1. Genaue Umschreibung oder Definition sowie zweckmäßige Verallgemeinerung eines vorgegebenen Problems.
2. Genaue Bestimmung aller die Lösungen des vorgegebenen Problems beeinflussenden Umstände, d.h. Bestimmung der Parameter des Problems.
3. Aufstellung des Morphologischen Kastens oder des vieldimensionalen Schemas, in dem alle möglichen Lösungen des vorgegebenen Problems ohne Vorurteile eingeordnet werden.

4. Analyse und Bewertung aller im Morphologischen Kasten enthaltenen Lösungen.

5. Wahl der optimalen Lösung und Weiterverfolgung derselben bis zu ihrer endgültigen Realisierung oder Konstruktion.

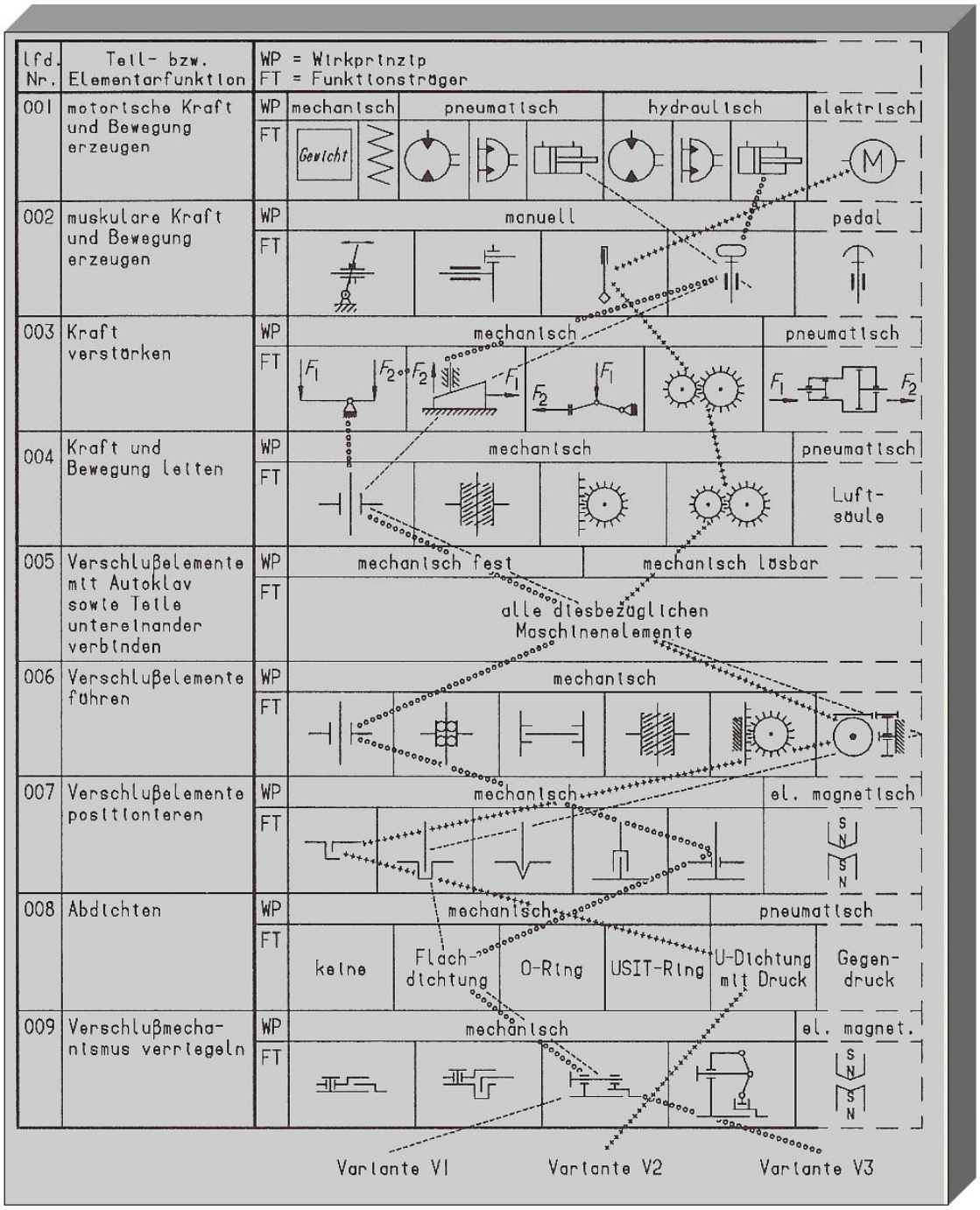


Abbildung 8: Beispiel eines Morphologischen Kastens für die Synthese eines Verschlussmechanismus eines Autoklavdeckels [Breiing/Flemming93, 55]

Es wird zu jeder Funktion bzw. Aufgabe *(Zeile)* eine geeignete Lösung *(Spalte)* ausgewählt. Die Auswahl wird durch die Angabe von Merkmalen unterstützt und durch Zusatzinformationen (z.B. Anhang) ergänzt. Die Betrachtung einer Zeile mit ihren Lösungsalternativen entspricht dem Nachschlagen in einem Konstruktionskatalog.

Die ausgewählten Lösungen der einzelnen Zeilen werden zu einer Gesamtlösung kombiniert und stellen eine Gesamtlösungsalternative dar. Durch die Wahl verschiedener Lösungsalternativen auf den einzelnen Funktionsebenen werden unterschiedliche Gesamtlösungsalternativen erzeugt. Eine Bewertung und Entscheidung für die optimale Alternative muss erfolgen.

Dreibholz [Dreibholz75] hat das Prinzip des Morphologischen Kastens verallgemeinert und generelle Ordnungsschemata für die methodische Konstruktion von Lösungen entwickelt. Er hat dabei drei ordnende Gesichtspunkte *Funktion, Lösung, Merkmal* erkannt. Mit diesen Ordnungsschemata lassen sich Lösungen für unterschiedlichste Anwendungsbereiche übersichtlich darstellen. Sie sind die Grundlagen für Konstruktionskataloge. Das Finden der richtigen Merkmale (Parameter) für eine Funktion und ihre Lösungen ist die Schwierigkeit beim Aufstellen eines Ordnungsschemas.

Dreibholz hat die Einsatzfelder für ein Ordnungsschema beim methodischen Konstruieren in [Dreibholz75, 236] folgendermaßen bestimmt:

1. Finden von Lösungen für Funktionen (kreatives Suchen).
2. Zusammenstellung der Lösungen von Teilfunktionen einer Gesamtfunktion (kreatives Suchen, Kombinieren zu Gesamtlösungen).
3. Überprüfen der Verträglichkeiten von Teillösungen für die Gesamtfunktion (kreatives Suchen, Beurteilen, Auswählen).
4. Bewerten von Lösungen (Beurteilen)
5. Übersicht über Merkmale von Lösungen, Objekten, Stoffen... (Darstellen)

Die Anwendung einer morphologischen Matrix steht folgender Schwierigkeit gegenüber: Prüfung der Verträglichkeit (Kombinationsfähigkeit) zwischen den Lösungen unterschiedlicher Teilaufgaben.

Roth [Roth94 Bd.1, 12ff] stellt die Problematik der vollständigen Kombination dar und zeigt auf, dass nie alle Lösungen miteinander kombinierbar sind. Durch einige Handlungsregeln unterstützen Pahl und Beitz [Pahl/Beitz93, 115ff] die Verträglichkeitsprüfung für den Konstrukteur. Dazu gehören:

- Die Teilfunktionen sollten in einer Funktionskette stehen. Eine sinnvolle, die Kombination unterstützende Reihenfolge ist notwendig.
- Die Lösungen mit Prinzipskizzen darstellen und die wichtigsten Merkmale (Eigenschaften) mit angeben. Der Morphologische Kasten wird mit den Informationen eines Konstruktionskataloges versehen.
- Es sollte ein schrittweises Konfigurieren erfolgen, wobei nur Verträgliches kombiniert werden darf. Die Anforderungen der Spezifikation (Pflichtenheft) sind immer auf Erfüllung zu prüfen. Optimale Kombinationen sollten hervorgehoben, analysiert und ihre Auswahl begründet werden.

Das Verträglichkeitsproblem ist mit einem Problem verknüpft, welches von Pahl und Beitz nicht ausdrücklich angesprochen wird:

- Kombinationskonflikte aufgrund mangelnder Baugruppenvariation

Werden komplexe Baugruppen in einer morphologischen Matrix als Lösungsalternativen aufgeführt, so können Kombinationskonflikte zwischen den Elementen der Baugruppen einer Teilaufgabe und den Lösungskomponenten von anderen Teilaufgaben bestehen. Ist keine Baugruppe ohne Konfliktelemente in den Alternativen enthalten, so werden nur wenig optimale Gesamtlösungen konfiguriert. Im Grunde ist dieses Problem als schlecht aufgespanntes Lösungsspektrum zu klassifizieren. Durch die Baugruppendarstellung ist die Gefahr jedoch wenig transparent. Es gibt nun zwei Möglichkeiten das Problem zu umgehen. Erstens: Es werden keine Baugruppen in morphologischen Matrizen verwendet. Damit würde allerdings ein großer Teil der Konstruktionsproduktivität verloren gehen. Zweitens: Für konfliktträchtige Elemente einer Baugruppe müssen möglichst viele Varianten in das Lösungsspektrum aufgenommen werden. Folglich steht dann eine größere Anzahl an Baugruppen-Varianten für die Kombinationsprüfung zur Verfügung.

2.1.4 Baukastensysteme, Varianten, Baureihen, Typengruppen

Eine modulare Bauweise von technischen Systemen hat sich bewährt. Es ergeben sich Vorteile in Bezug auf die Wiederverwendung und Arbeitsteilung in der Konstruktion sowie Losgrößenvorteile und Möglichkeiten zur Fremdherstellung in der Fertigung. Das spiegelt sich auch im Arbeitsschritt 4 des generellen Vorgehensmodells wieder. Die Ebenenbildung für die Zerlegung von Aufgaben in Teilaufgaben und die entsprechende Komposition von Teillösungen zu Gesamtlösungen ist natürlich vielschichtiger als in Abbildung 4 dargestellt. Durch die vielen verschiedenen Schichten lassen sich auf vielen Abstraktionsebenen möglicherweise direkt Lösungsbaugruppen finden, die für die Konstruktion zu verwenden sind. Die modulare Bauweise verkürzt somit den Entwicklungsprozess für Produkte. Voraussetzung für eine modulare Bauweise sind definierte Schnittstellen (Flansch-, Pass- oder Anschlussstellen) zwischen den Komponenten [Koller85, 112]. Eine vollständige Austauschbarkeit der Bauteile wurde erst durch die Normierung der Schnittstellen und der Entwicklung eines standardisierten Toleranzsystems ermöglicht [Krause93, 62ff]. Für die Anwender von modularen Systemen ergeben sich Vorteile in Bezug auf Wartungs- und Ersatzteilkosten sowie die Möglichkeit, bei Erweiterung und Austausch auf andere Hersteller zurückgreifen zu können. Dies funktioniert allerdings nur, wenn die Schnittstellen herstellerübergreifend standardisiert sind.

Verfolgt man das Ziel, mit möglichst wenigen unterschiedlichen Bauteilen möglichst viele verschiedene Gesamtsysteme zu entwickeln, so ist ein Baukastensystem erforderlich.

Definition:

> Ein **Baukastensystem** ist eine Anwendung der Kombinationsgesetze auf zusammengesetzte Gegenstände mit dem Ziel, die Haupteigenschaften der Kombinationen auszunutzen, nämlich mit einer kleinen konstanten Anzahl verschiedener Elemente eine große Anzahl Komplexionen zu bilden [Nasvytis53, 86].

Die Elementarisierung von Gegenständen ist die Grundlage zur Entwicklung eines Baukastens. Dies ist ein Normungsvorgang [Borowski61, Nasvytis53]. Es ist dabei nicht vorgeschrieben, dass alle Elemente eines Baukastens real existieren müssen, die

„Bauteile“ können ebenso gut nur aus „Konzepten“ (Beschreibungen) bestehen [Pahl/Beitz93, 596ff].

Ein Baukastensystem ist dadurch gekennzeichnet, dass der gleiche Bausteintyp in einem Gesamtsystem mehrmals eingesetzt werden kann und dass an einer Stelle eines Systems wahlweise verschiedene Bausteine eingesetzt werden können [Koller85, 114].

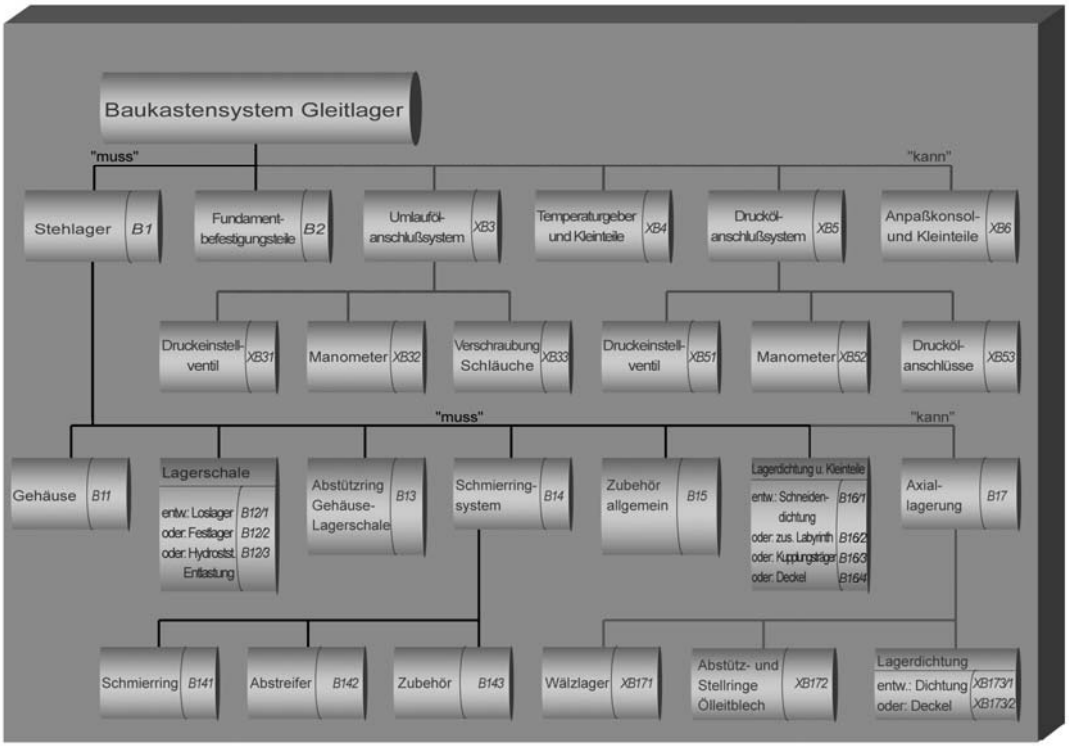


Abbildung 9: Beispiel eines Baukastensystems für Radialgleitlager [vgl. Pahl/Beitz93, 603]

Viele Baukastensysteme versorgen einige wenige Gleichteile mit vielen (frei) kombinierbaren variablen Teilen. In dem Leitfaden zur Erstellung eines maschinellen Variantensuchsystems [DIN Variantenübersicht86] wird die Variantenerzeugung anhand eines Fahrradbeispiels gezeigt. Deutlich wird dabei, dass Produktfamilien entstehen, die aufgrund wesentlicher Merkmale eine Benennung erhalten (z.B. Sportfahrräder). Es wird allerdings ebenfalls ersichtlich, dass sich auch die Gleichteile variieren lassen, dadurch ist der Unterschied zwischen Gleichteilen und Baukasten-teilen nicht eindeutig zu definieren.

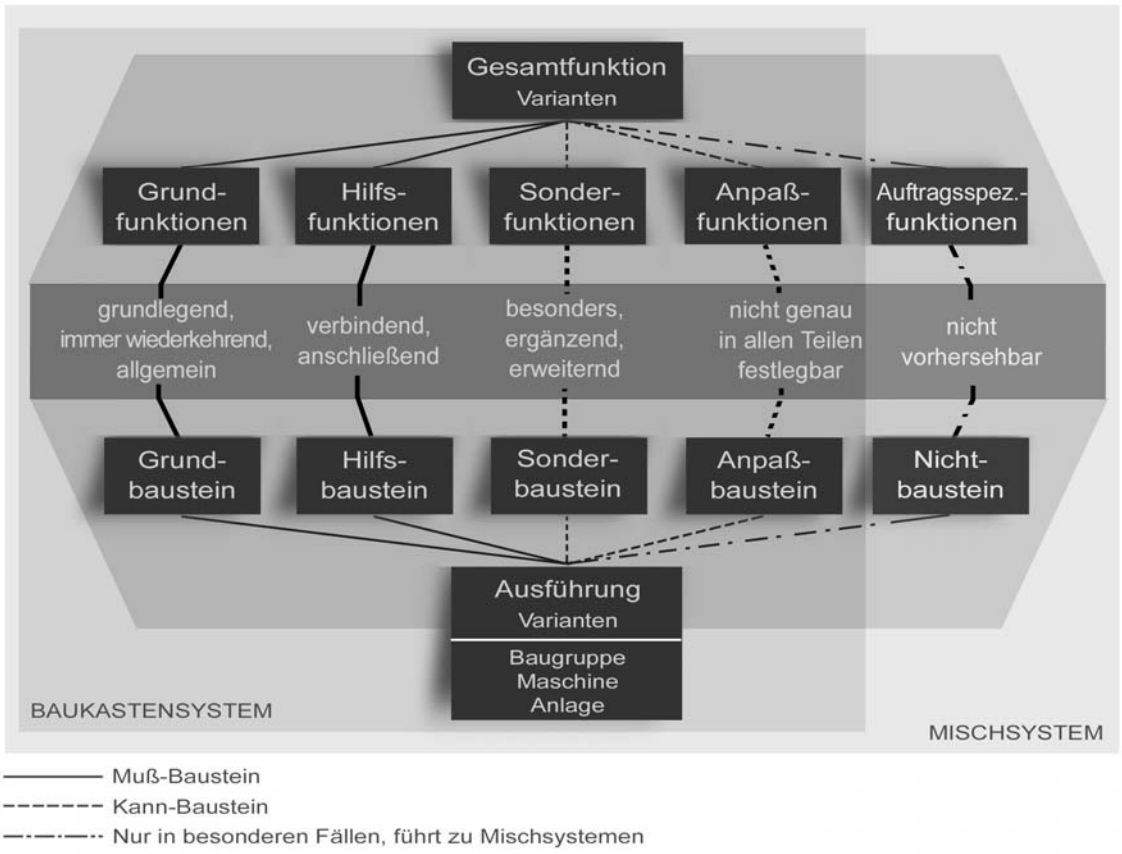


Abbildung 10: Bausteinarten für Baukasten- und Mischsysteme [vgl. Pahl/Beitz93, 592]

Die Vorteile von Baukastensystemen sind in der effizienteren Konstruktion zu sehen und der Möglichkeit, Veränderungen des Gesamtsystems durch Austausch oder Erweiterung einzelner Komponenten zu realisieren. Nachteile liegen in der eingeschränkten Fähigkeit, ein Gesamtsystem auf eine spezifische Aufgabe hin zu optimieren. Ein Baukastensystem wird nicht dafür entwickelt, um für jeden Kunden (Anwender des Baukastens) wieder neue Bauteile zu entwickeln.

Die Baukastensysteme der Technik haben für diesen Fall nicht vorhersehbare, auftragsspezifische „Nicht-Bausteine“ definiert, die jedoch aus einem Baukastensystem ein Misch-System machen [Borowski61, Breiing/Flemming93, Pahl/Beitz93].

Natürlich werden durch die Anwendung des Baukastensystems permanent Verbesserungsvorschläge seitens der Anwender entstehen und in die laufende Weiterentwicklung des Baukastensystems einfließen. Zudem werden sich die Anwendungsbereiche und der Funktionsumfang eines Baukastensystems laufend vergrößern, so

dass eine Erweiterung des „Produktangebotes“ ebenfalls durch die Vermehrung von Bausteinen realisiert wird. Dabei spielt es aus der Sicht des Anwenders eines Baukastensystems keine Rolle, ob die Bausteine von einem oder von vielen Herstellern stammen. Das Vertrauen der Kunden in „Zukauf-Baukastensysteme“ setzt jedoch eine entsprechende Zertifizierung voraus.

Eine geplante Variation ist für den Aufbau eines Baukastensystems besonders wichtig. Sind erst die Teilfunktionen (Teilaufgaben) identifiziert, deren Lösungen variiert werden sollen, ist es möglich die Bausteinvarianten systematisch zu erzeugen.

Die Baureihen- und Typengruppen-Variation der Bausteine ist ein Mittel zur Rationalisierung der Konstruktion und Fertigung [Koller85, 114]. Baureihen werden durch die Variierung eines Konstruktionsparameters in festgelegten Abstufungen (in einer Reihe) gebildet. Beispielsweise sind dies Größenbaureihen, Gestaltbaureihen, Abmessungs – und Volumenbaureihen und weitere. Daneben können Parameter systematisch variiert werden, deren Werte keiner Reihe entsprechen, sie werden bei Koller [Koller85, 114ff] Typengruppen genannt. Beispiele hierfür sind Werkstoffe, Qualität, Form und andere. Jeder Parameter eines technischen Systems kann beliebig viele Werte annehmen, in der Praxis sind jedoch nur relativ wenige Werte von Interesse. Aus diesem Grund werden nur wenige diskrete Werte für die Variation zugelassen und als Baureihen oder Typengruppen standardisiert [Koller85, 114f]. Ein Erzeugen von Lösungsalternativen durch ihre Parameter-Werte-Variation entspricht der Morphologischen Methode von Zwicky [Zwicky71].

Die begriffliche Unterscheidung zwischen Varianten, die nicht Baureihen sind, und Baureihen bzw. Typengruppen ist abhängig von den Variationen der Merkmale eines Bausteintyps.

Ein Glühlampen-Beispiel soll die Situation verdeutlichen:

Gegeben sind:

- drei Formen („Kerze“, „Birne“, „Kugel“) und
- vier Leistungsstufen (40, 60, 80, 100 Watt) sowie
- zwei Glaseigenschaften (matt, klar).

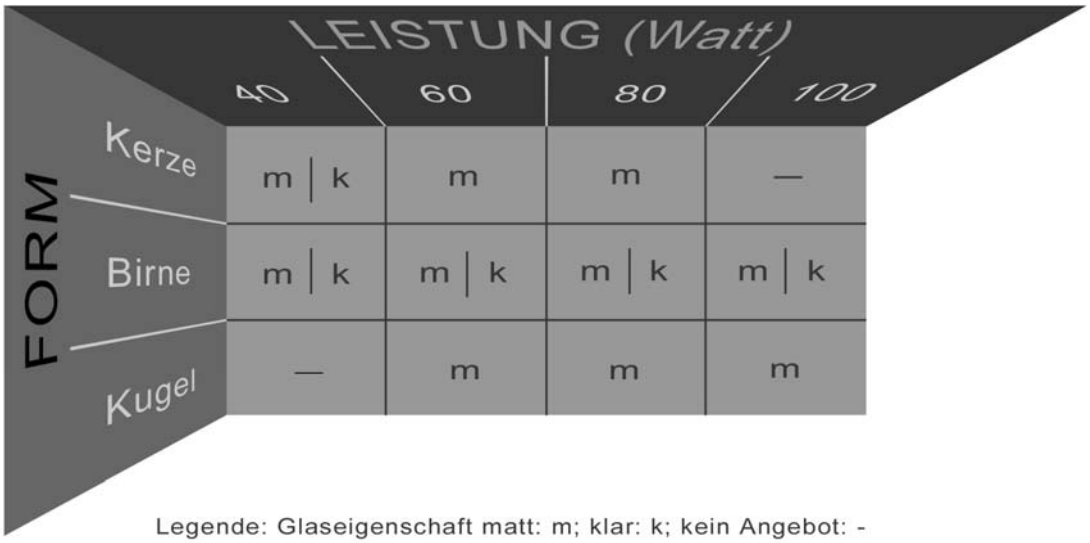


Abbildung 11: Matrix von Glühlampeneigenschaften und Sortimentangebot

Gemäß Kollers Festlegung ist jede Glühlampenform, wird die Leistungsaufnahme betrachtet, Element einer Baureihe (Leistungsbaureihe). Jede Glühlampe einer Leistungsklasse ist, über die Variation ihrer Form, Element einer Typengruppe (Formtypengruppe). Die alternativen Glaseigenschaften, die in dem o. a. Sortiment nicht für jede Ausführung variieren, sind unspezifische Varianten. Es existieren noch weitere Baureihen der Glühlampen bezüglich ihrer Eingangsspannung (140, 220 Volt), die jedoch in der Auswahl hier nicht von Interesse sind. Dadurch soll verdeutlicht werden, dass Bausteine auch noch über versteckte Merkmale variiert werden können. Die Ausprägungen dieser versteckten Merkmale müssen vor der Sortimentsentnahme schon festgelegt sein oder eine explizite Wertangabe muss noch erfolgen.

In der Konstruktionslehre wird unter Variante entweder der generelle Begriff Variante ($Variante_G$) mit nachfolgender Definition verwendet oder die Variante dient der Abgrenzung zur Baureihe.

Definition der $Variante_G$:

> **Varianten**, sind Bausteintypen, die alle denselben Schnittstellen-bedingungen genügen und für die Realisierung einer Teilaufgabe unterschiedliche Eigenschaften haben.

Im Gegensatz zur Baureihe bedingt die Variante eine nicht geordnete Variation eines oder mehrerer Konstruktionsparameter (Eigenschaften). Die Baureihe dagegen ist definiert durch die Variation nur eines Parameters. Wobei die Zuordnung eines Bausteintyps zu einer Baureihe relativ ist, denn er kann durchaus mehreren unterschiedlichen Reihen angehören. Uneinheitlich ist nun die Art der Variation der Parameter einer Baureihe. Im Gegensatz zu Koller wird oft nicht nach Typengruppen und Baureihen („Baureihe$_S$“) unterschieden. Oder die Typengruppen fallen unter den spezialisierten Begriff „Variante“. In der vorliegenden Arbeit wurden die Begriffe „Variante“ und „Baureihe“ folgendermaßen festgelegt (Begriffsverwendung in Grossbuchstaben):

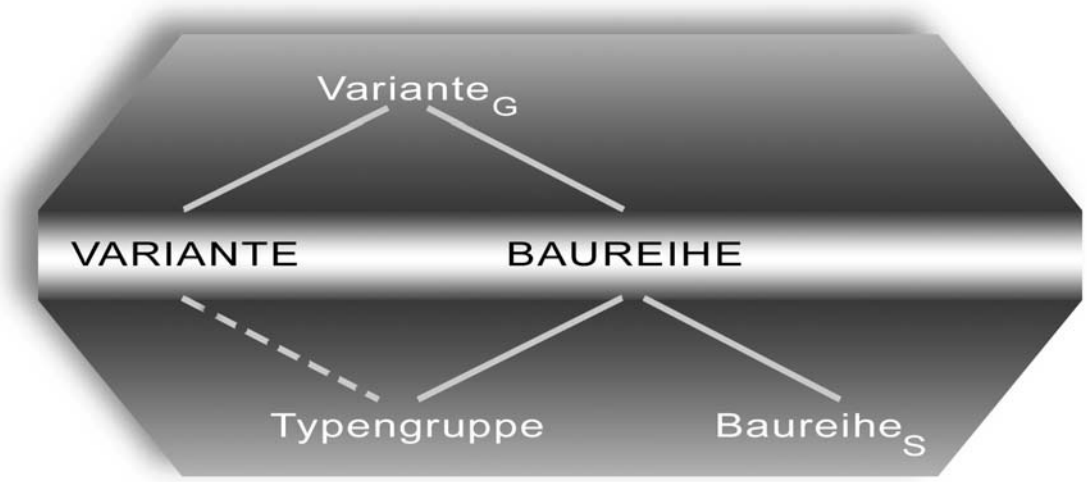
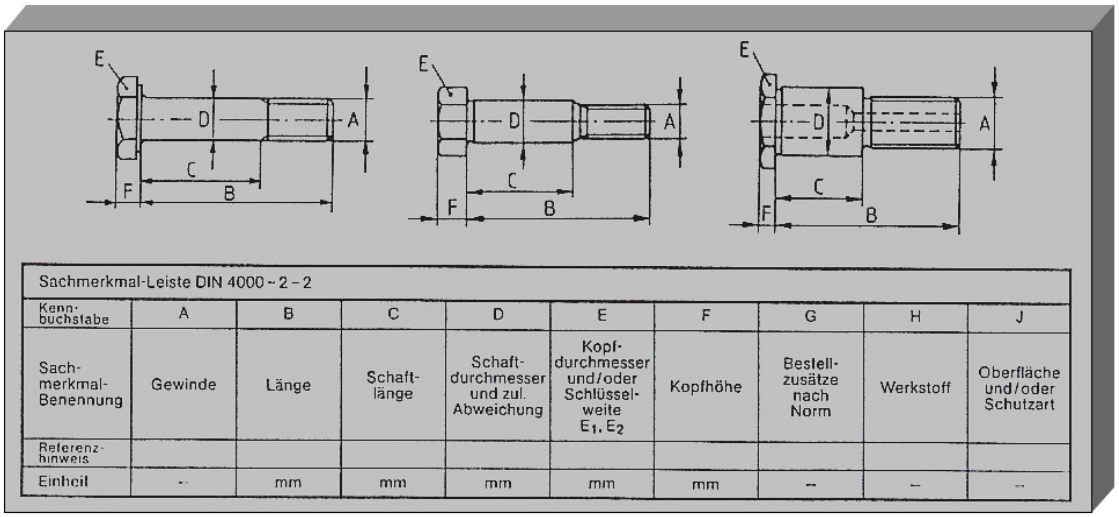


Abbildung 12: Begriffsvarianten von „Variante“ und „Baureihe“

2.1.5 Sachmerkmalleisten

Eine strukturierte Bereitstellung der Zugriffsmerkmale für die Gegenstände von Lösungskatalogen und Baukastensystemen kann durch Sachmerkmalleisten erfolgen. Sachmerkmale dienen der Beschreibung und Identifikation von Gegenständen durch charakterisierende Eigenschaften, dargestellt durch Merkmale und ihre Ausprägungen. Sachmerkmalleisten fassen ähnliche Gegenstände über gemeinsame Merkmale zu Klassen zusammen [Meinl90, DIN 4000-92, Krauser86]. Obwohl die Merkmalausprägungen der Sachmerkmalleisten Bestandteil der Produktdokumentation sind [DIN 4000-92], ist es nicht ihr erklärtes Ziel, Gegenstände vollständig zu beschreiben bzw. die Spezifikationen für Herstellung und Abnahmeprüfung zu liefern.

Sachmerkmalleisten repräsentieren eine zweidimensionale Darstellung eines (Gegenstand-) Sortiments eines gegebenen Anwendungsbereiches. Folgende Abbildung soll durch ein Beispiel einer Sachmerkmalleiste das Prinzip noch etwas verdeutlichen.



Sachmerkmal-Leiste DIN 4000 – 2 – 2

Kennbuchstabe	A	B	C	D	E	F	G	H	J
Sachmerkmal-Benennung	Gewinde	Länge	Schaftlänge	Schaftdurchmesser und zul. Abweichung	Kopfdurchmesser und/oder Schlüsselweite E_1, E_2	Kopfhöhe	Bestellzusätze nach Norm	Werkstoff	Oberfläche und/oder Schutzart
Referenzhinweis									
Einheit	–	mm	mm	mm	mm	mm	–	–	–

Abbildung 13: Beispiel einer Sachmerkmalleiste für Passschrauben nach [DIN 4000-93][8]

Ein technisches Ordnungssystem kann in drei Ebenen eingeteilt werden:

- Sachmerkmalleisten für die Eigenschaftsverwaltung
- Konstruktionskataloge für die Lösungsverwaltung
- Baukastensysteme für die Variantenverwaltung

Wobei diese Ebenen in einer Reihenfolge anzuordnen sind: Die Sachmerkmalleisten dienen dem Zugriff auf die Elemente der Konstruktionskataloge. Konstruktionskataloge unterstützen den Zugriff auf Elemente von Baukastensystemen.

[8] Die graphischen Erläuterungen der Kennbuchstaben stellen die Worterklärungen einer normierten Terminologie dar.

2.1.6 Konstruktionsarten

Vorgehen und Aufgaben in der Konstruktion variieren, abhängig von den Konstruktionsbereichen bzw. vom Konstruktionsanlass. Es lassen sich folgende Grundtypen unterscheiden [VDI 2222-97, 7]:

- Entwicklungskonstruktion: vorwiegend bei auftragsunabhängiger Produktion, aber auch bei grundsätzlicher (prototypischer) Überarbeitung auftragsabhängig produzierter Erzeugnisse, meist werden alle wesentlichen Konstruktionsmerkmale überarbeitet.
- Auftragskonstruktion: wird durch Kundenaufträge initiiert, die Realisierung erfolgt meist auf dem Wege der begrenzten bzw. lokalen Anpassung bekannter, i.Allg. früherer Lösungen.
- Angebotskonstruktion: erfolgt auf Kundenanfragen hin, sie hat i.Allg. projektierenden Charakter, baut meist auf früheren Lösungen auf und dient als Basis für die Angebotskalkulation. Sie hat gelegentlich die Funktion einer Machbarkeitsstudie.

Nach Steinhilper und Röper sind Konstruktionsprozesse kreative Vorgehensweisen. Dadurch ist eine weitere Unterscheidung nach der Intensität des kreativen Einsatzes zum Konstruktionszeitpunkt möglich. Die Grenzen dieser verschiedenen Arten der Konstruktionstätigkeit sind fließend [Steinhilper/Röper94, 7]:

- Neukonstruktion: Erstellen einer neuen Lösung für ein System bei gleicher oder veränderter Aufgabenstellung.
- Anpassungskonstruktion: Anpassen eines vorliegenden Systems an eine veränderte Aufgabenstellung. Das grundsätzliche Lösungsprinzip bleibt erhalten, verändert werden nur Teilsysteme, um die geforderte Funktion zu erbringen.
- Variantenkonstruktion: Anpassen der Systeme in Größe (Baureihen) und Anordnung (Baukasten) innerhalb der Anwendungsgrenzen. Funktion und Lösungsprinzip bleiben gleich.

Die Variantenkonstruktion erfordert den geringsten Kreativitätsaufwand. Dem ist jedoch der Aufwand für die einmalige Neukonstruktion zur Erstellung der Elemente-Varianten (z.B. ein zu Grunde liegendes Baukastensystem) gegenüberzustellen [Pahl/Beitz93].

Die Variantenkonstruktion stellt eine Teilmenge aller Konstruktionsschritte einer Neukonstruktion dar [Breiing/Flemming93, 7]. Das Wesentliche der Variantenkonstruktion ist, dass auf schon bestehende Komponenten zurückgegriffen wird, diese entweder neu konfiguriert werden (Variantenkonfiguration) oder eine Abwandlung als Variante neu konstruiert und hergestellt wird (Variantenherstellung). In umfangreichen Konstruktionsprojekten ist die Anwendung aller Konstruktionsarten möglich.

Existieren umfangreiche Bauteilesammlungen in Form von Baukastensystemen für die Variantenkonstruktion, so ist die zusätzliche Nutzung von Katalogsystemen für das Retrieval innerhalb der Baukastensysteme sinnvoll. Die Kombination von Variantenkonstruktion und Katalogkonstruktion verspricht eine sehr hohe Effizienz in Bezug auf die Konstruktionstätigkeit und die Fertigung. Zum einen verkürzt die Katalogkonstruktion den Konstruktionsvorgang auf das Finden der geeigneten Teilaufgabe und die Auswahl der geeigneten Lösung zu dieser Teilaufgabe [VDI 2222-82] und zum anderen kann die Variantenkonstruktion das Finden der Teilaufgaben verbessern und unterstützt durch eine geeignete Baukastenkonstruktion die Auswahl der besten Lösung.

2.1.7 Zusammenfassung

Ott hat die Vorgehensweise eines Ingenieurs basierend auf einer Interviewserie mit Ingenieuren und auf Veröffentlichungen des VDI auf folgende Prinzipien reduziert [Ott94, 32ff]:

1. **Systematisches Vorgehen**
   Vorgehen nach einer Methode mit Unterstützung von methodenspezifischen Werkzeugen.

2. **Denken in Baugruppen**
   Ein strukturiertes Vorgehen erfordert die Unterteilung des Gesamtsystems in Subsysteme (Baugruppen) und unterteilt diese wiederum in Subsysteme (Bauteile oder Baugruppen) solange, bis die erhaltenen Subsysteme „technisch handhabbar“ sind. Die „Strukturierung

erfordert ein Denken in Systemzusammenhängen, um die Subsysteme wieder zum Gesamtsystem *integrieren* zu können.“ Ott benennt ausdrücklich die Moduleigenschaft von Baugruppen, wonach sie in sich abgeschlossen und abgrenzbar sind und eine definierte Funktionalität aufweisen. Für die Integration zum Gesamtsystem müssen sie geeignete Schnittstellen bereithalten.

3. **Wiederverwendung**
   Bereits vorhandene Baugruppen sollten nach Möglichkeit nicht nochmals entwickelt, sondern wieder verwendet werden. Diese Baugruppen sind häufig bereits Normteile (de facto bzw. von Normierungsgremien), die teilweise in CAD-Normteil-Katalogen zur Verfügung gestellt werden.

4. **Prozessstrukturierung**
   Das strukturierte Denken wird nicht nur auf das Produkt, sondern auch auf das Produktionsverfahren (einschließlich der Entwicklung) angewandt.

5. **Prozessbegleitendes Qualitätsbewusstsein**
   Jedes Teilergebnis im Entwicklungsprozess wird von Spezialisten erbracht, die entsprechende Fähigkeiten besitzen, die Teilergebnisse in hoher Qualität zu erstellen. Jedes Teilergebnis muss so erstellt werden, dass es einer Qualitätskontrolle unterzogen werden kann.

Das Denken in Baugruppen ist die eigentliche Konstruktionsleistung. Die wesentliche Herausforderung dabei ist der richtige Umgang mit Abstraktions- und Komplexionsebenen. Damit die geeigneten Baugruppen identifiziert werden können, muss die Aufgabenstellung sowohl in Bezug auf die Anforderungen an das System (Zweckbeschreibungen und Bedingungen) als auch in Bezug auf Anforderungen an die Konstruktion und Herstellung (Produktivität) korrekt erfasst sein. Die Produktivität wird durch Maßnahmen zur Verkürzung der Konstruktion beeinflusst. Dabei wirken folgende Konstruktionsmethoden optimierend:

- Entwickeln eines Baukastensystems
- Konstruieren mit Lösungskatalogen
- Variantenkonstruktion

Für produktivitätssteigernde Maßnahmen ist einerseits eine strenge Modularisierung und andererseits der Wille zur Wiederverwendung notwendig. Für beide ist das Finden der geeigneten Teilaufgaben, also einer zweckmäßigen Baugruppen-Begrenzung, eine Voraussetzung. Ist diese Aufgabe erfolgreich erledigt, steht noch die richtige Variation der Lösungen an, d.h. ein möglichst vollständiges Aufstellen sinnvoller Lösungsalternativen.

Die vielen Einzelaufgaben in der Konstruktion werden durch ein ausgereiftes Methodenspektrum unterstützt. Auf die Konstruktionshandlungen Abstraktion und Komposition sowie die Morphologische Methode zur systematischen Lösungsfindung wurde hier eingegangen. Eine Übersicht verschiedener Methoden zu den einzelnen Konstruktionsschritten ist in der Methoden-Matrix in [VDI 2221-93] zu finden.

2.2 Konstruktionstheorie für betriebswirtschaftliche Standard-Anwendungssysteme

Seit über 35 Jahren wird die legendäre Software Engineering Konferenz in Garmisch im Jahre 1968 zitiert, bei der die Wünsche nach ingenieurgemäßen Konstruktionsprinzipien für komponentenbasierte, industriell gefertigte Softwaresysteme formuliert wurden [McIlroy69]. Allerdings hat sich dieser Wunsch noch immer nicht erfüllt. Aus diesem Grund wird der vorliegende Abschnitt nicht mit Konstruktionslehre tituliert, sondern mit Konstruktionstheorie. Bei der Betrachtung von Konstruktionsmethoden für Standard-Anwendungssysteme ist es wichtig zu verstehen, dass die Herstellung von Standard-Anwendungssystemen von der Entwicklung von Systemen für einen bestimmten Einsatzzweck (Individualsoftware) unterschieden werden muss.

Die Codebasis von Standard-Anwendungssystemen muss, nach Abwägung wirtschaftlicher Erfordernisse, permanent an Technologiewechsel angepasst werden. Das jeweils beim Kunden installierte Anwendungssystem kann Technologieveränderungen ignorieren, der Hersteller muss sich jedoch immer den aktuellen Markttrends anpassen. Andernfalls ist kein Neukundengeschäft bzw. kein Upgradegeschäft mehr möglich. Da aus Wartungseffizienzgründen keine separate Codebasis für die jeweiligen Technologiestufen vertretbar ist, müssen Standard-Anwendungssysteme möglichst technologieunabhängig konstruiert werden. Einige Beispiele für Technologie-Entkopplungsbereiche sind:

- Betriebssystem
- Datenbank-Management-System
- Dokumentenaustausch mit Partnern und Fremdsystemen
- Formulardruck
- Berichtswesen
- Fremdsystemintegration
- Dokumentenablage
- Kollaborationstechniken (Marktplätze, e-Commerce)
- Telefonie-Techniken (CTI)
- Technologie für Benutzeroberflächen (GUI-Bibliotheken, Portale, Sprachintegration)

Standard-Anwendungsysteme erhöhen ihre Komplexität in der Systemarchitektur um diesen Entkopplungsbedürfnissen gerecht zu werden. Zudem sind viele Entwicklungswerkzeuge, die am Markt verfügbar sind, in diesen Bereichen auf eine Technologie festgelegt und somit nicht für den Einsatz für Standard-Anwendungssystementwicklung geeignet.

Von Standard-Anwendungssystemen wird eine hohe betriebswirtschaftliche Integration erwartet [Hufgard et.al. 05]. Denn gerade die betriebswirtschaftliche Integration ist sehr aufwändig zu entwickeln und nur Hersteller mit einer hohen Verkaufsstückzahl sind in der Lage diese hohen Investitionskosten zu leisten.

Ein mehrstufiges Fertigungsprinzip ist eine weitere Grundlage dafür, dass Standard-Anwendungssysteme erfolgreich sein können. Die Globalisierung zwingt jeden Hersteller in ein Konzept verteilter Entwicklung. Somit muss ein Fertigungskonzept aufgebaut werden, welches eine zentrale Entwicklung für Kernfunktionalität sowie eine dezentrale, global verteilte Entwicklung für industrie- und länderspezifische Zusatzentwicklung ermöglicht. Zudem müssen Entwicklungspartner für bestimmte Marktsegmente mit in die Fertigungskette eingebunden werden können, damit eine entsprechende Marktfokussierung für kleinere Zielmärkte geleistet werden kann. Das Anwendungssystem-Produkt, welches beim Unternehmen des Kunden dann installiert wird, sollte dennoch für die Benutzerinteraktion, Dokumentation, IT Administration und

Wartung wie aus „einer Hand“ entwickelt aussehen. Für den kundenindividuellen Einsatz kommt häufig noch eine letzte Fertigungsstufe hinzu, die Anpassungsentwicklung. Auch dieser Umstand stellt einen gewichtigen Unterschied zu Individualsoftware dar, denn dort ist Anpassungsentwicklung Teil des Neuentwicklungsprojektes, wohingegen bei Standard-Anwendungssystemen Techniken zur möglichst „modifikationsfreien“[9] Anpassungsentwicklung eingesetzt werden müssen, damit ein Release-Wechsel für die Kunden wirtschaftlich vertretbar bleibt.

Damit für die Weiterentwicklung von Standard-Anwendungssystemen auf der Seite des Herstellers so wenig Einschränkungen wie möglich geschaffen werden, haben sich für die letzte Fertigungsstufe beim Kunden spezielle Techniken zum Einbinden von kundenindividuellen Programmen etabliert. Dies sind beispielsweise Datenaustausch-Tabellen oder ausgewählte veröffentlichte Funktionsschnittstellen bzw. Erweiterungspunkte, die von Programmroutinen aufgerufen werden (beispielsweise SAP User Exits). Diese bieten den Kunden Investitionssicherheit und auf Herstellerseite stellen sie die Weiterentwicklung sicher.

Um bezüglich der mehrstufigen Fertigung mit der technischen Konstruktionslehre vergleichbar sein zu können, müssten die Konstruktionsmethoden über alle Fertigungsstufen hinweg dieselben sein. Dies ist im Bereich der Standard-Anwendungssysteme noch nicht der Fall.

Standard-Anwendungssystem-Entwickler müssen bei ihren Konstruktionshandlungen immer viele verschiedene Kundenunternehmen mit ihren unterschiedlichen betriebswirtschaftlichen Szenarien zur Geschäftabwicklung im Blick haben. D.h. für Standard-Anwendungssysteme werden alternative Funktionen, Prozesse, Datenstrukturen und Ausgabeformen konstruiert, die über Konfiguration und Erweiterung für einen speziellen Kunden angepasst werden. Erweiterung heißt, in vielen Bereichen werden Möglichkeiten für kundenindividuelle Zusatzentwicklung (Fertigungsstufe „Kunde“) geschaffen. Je größer der globale Zielmarkt für das Standard-Anwendungssystem ist, desto mehr Variabilität und damit auch Komplexität muss in das Standard-Anwendungssystem hineinkonstruiert werden. Diese erhöhte Variabilität wird zusehends mit „Multifunktionsmaschinen“ realisiert. Beispielsweise wird nicht eine Auftragsabwicklung für

[9] Modifikationsfrei heißt, dass kundenindividuelle Entwicklungsergebnisse auch nach einem Release-Wechsel des Herstellers – ohne Anpassungsaufwand – weiterhin funktionieren.

Lagerprodukte und eine andere Auftragsabwicklung für Serviceprodukte parallel entwickelt sondern es werden Programme zur Verarbeitung aller Auftragsarten entwickelt. Folglich werden die Konfigurationsaufgaben immer aufwändiger und sind schwieriger zu überblicken.

In den nachfolgenden Ausführungen wird beginnend mit der relationalen Datenmodellierung, ein historischer Überblick über die Konstruktionsansätze zum Bau von Standard-Anwendungssystemen gegeben.

Entity Relationship Modellierung

Als Chen [Chen76] eine Konstruktionsmethodik für relationale Datenbanken entwarf, war die Grundlage gelegt für eine ingenieurgerechte Entwicklung von Standard-Anwendungssystemen (hier: Datenbank-Anwendungen). Systemarchitekten waren nun in der Lage für die Objekte und Beziehungen der realen Welt Repräsentanten als Entities (Entitäten) und Relationships (Beziehungen) in einem Datenmodell grafisch abzubilden. Batini u.a. haben nachfolgend aufgeführte Qualitätskriterien für Datenmodelle aufgestellt: vollständig, korrekt, minimalistisch, ausdrucksvoll (ausdrucksstark), lesbar, selbsterklärend, erweiterbar und normalisiert [Batini et.al. 92, 139ff]. Werden diese Qualitätskriterien eingehalten, dann ist ein konzeptuelles Datenmodell als Konstruktionszeichnung für die Datenstruktur eines Anwendungssystems zu betrachten. Ein weiteres Prinzip der ingenieurgerechten Konstruktion ist die Modulbildung [Hruschka86]. Zur konsequenten Anwendung des Modulprinzips auf die Anwendungsprogrammstrukturen, ist eine Datenintegration, basierend auf einem relationalen Datenbankschema (Datenmodell physikalisch abgebildet in einem Datenbank-Management-System), die ideale Unterstützung.

Die Verwendung eines integrierten Datenbankschemas unterstützt die Konstruktionshandlungen für die unterschiedlichen Geschäftsabwicklungsstrategien der potenziellen Kunden eines Standard-Anwendungssystems. Denn die verschiedenen Anwendungsszenarien können anhand des Datenmodells durchgespielt werden. Zudem wird eine verteilte und mehrstufige Entwicklung ermöglicht, da sich alle Parteien mit ihrer Entwicklung immer auf das Datenbankschema stützen.

Auf Modellierungstechniken zur Unterstützung der Konstruktionstätigkeit soll hier nicht näher eingegangen werden. Es werden zwei Aspekte für Modellierung in der Softwareindustrie erwähnt:

1. Ausdruckstarke versus ausdrucksarme Modellierungssprachen
2. Werkzeugunterstützte Modellierungssprachen versus nicht werkzeugunterstützte Modellierungssprachen

Ad 1) Dieser Punkt betrachtet den Aspekt, dass es unterschiedlich ausdrucksstarke Modellierungssprachen gibt. Beispielsweise hatte Martin für die Datenmodellierung mit seiner „Krähenfuss-Methode“ großen Erfolg in industriellen Projekten, denn die Methode ist leicht verständlich, da es wenig Sprachkonstrukte gibt [Martin89]. Es entsteht der Eindruck „jeder kann diese Sprache lernen, jeder versteht sie“. Auf der anderen Seite gibt es ausdrucksstarke, reichhaltige Sprachen, wie beispielsweise die Objekttypenmethode für eine semantische Datenmodellierung [Ortner/Söllner89]. Semantisch reichhaltige Modellierungssprachen zielen auf eine möglichst ganzheitliche Informationsmodellierung im Datenmodell. Wohingegen sprachlich enger gefasste Entity Relationship Diagramme in konkreten Softwareentwicklungsprojekten ohne viel Aufwand die wesentlichen Diskussionspunkte skizzieren können. Allerdings bleiben bei den ausdrucksarmen Modellierungssprachen viele ungelöste Designfragen offen, die dann zu umfangreichen Interpretationsspielräumen in der Phase der Implementierung führen.

Ad 2) Dieser Aspekt soll darauf hindeuten, dass es ohne adäquate Werkzeugunterstützung keinen Markterfolg für Modellierungstechniken gibt. Nur durch eine Werkzeugunterstützung in der Entwicklungspraxis werden auch die Methoden akzeptiert. Mit der UML (Unified Modelling Language) hat sich auch für die Datenmodellierung eine Modellierungssprache mit entsprechender Werkzeugunterstützung durchgesetzt [UML04].

Dies führt zu folgendem Schluss:

- Ganzheitliche Konstruktionsmethoden mit Werkzeugunterstützung werden den größten Erfolg haben.

Damit nicht immer wieder die gleichen Programmkonstrukte erstellt werden müssen, um die Daten aus der Datenbank auszulesen und zu verarbeiten, haben sich in den

Standard-Anwendungssystemen Funktionsbibliotheken gebildet, die dann für verteilte und mehrstufige Fertigungsprozesse als Konstruktionselement zur Verfügung stehen. Dies war ein weiterer Schritt in Richtung industrielle Konstruktions- und Fertigungsprinzipien.

Objektorientierung

Der nächste Entwicklungsschritt war, die funktionalen und datenzentrierten Aspekte der Systemarchitektur in einem Modell zu vereinen. Die Objektorientierung löst genau dieses Problem [Coleman et.al. 94, Goldberg/Rubin95, Meyer96]. Zudem wird die Objektorientierung dem Anspruch Sachverhalte der Wirklichkeit im System mit denselben Begriffen, Eigenschaften und Verhalten wie in der realen Welt abzubilden besser gerecht als die Methoden zuvor. Es wird versucht, die Objekte des Anwendungsbereiches als Objekte der Software abzubilden. Objektorientierte Systeme entwickeln sich sehr schnell zu Programmen, die auf der Entwicklungsebene aus vielen Objekten mit komplexen Vererbungshierarchien bestehen, die zwar eine verbesserte Wiederverwendung von entwickelten Unterfunktionen sicherstellen, jedoch nicht für eine Konstruktionsmethodik, die sich auf den Anwendungsbereich abstützt und umfangreiche Systeme beherrschbar strukturiert, geeignet ist [Broy/Siedersleben02]. Die Funktionsbibliotheken wurden um Methodenbibliotheken für Standard-Anwendungssysteme erweitert. Die relationalen Datenmodelle existieren weiterhin, denn Objekte in Standard-Anwendungssystemen speichern ihre Daten immer noch in relationalen Datenbanken ab.

Application Frameworks

Mit dem Architekturprinzip der Application Frameworks[10] konnte das Prinzip der Vererbung um das Prinzip der Komposition erweitert werden. Dabei ging man von der Idee aus, dass ein komplexes objektbasiertes System im Zeitverlauf für eine bestimmte Domäne (Anwendungsbereich) weiter entwickelt wird. In diesem Framework werden zentrale Funktionalitäten entwickelt. Für die Erweiterung und Variation durch Verwender des Frameworks werden an bestimmten Stellen so genannte Hot spots

[10] Ein Framework „embodies the invariant part of a domain analysis“ [Coleman et.al. 94, 219].

bereitgestellt [Pree97, 63ff, Schmid et.al. 95, 43ff]. Das Framework ruft über eine Einschubtechnik die Erweiterungs- oder Variationsbausteine auf. Allerdings behält das Framework dabei immer die Programmsteuerung in seiner Kontrolle. Ursprünglich war dieser Ansatz dafür gedacht, umfangreiche Business Anwendungen für vielfachen Einsatz in Unternehmen zu entwickeln. Jedoch hat es sich gezeigt, dass die Konstruktion von Frameworks für ein komplettes Anwendungssystem viel zu komplex wird und zudem die Erweiterung ein tiefes Verständnis der framework-internen Programmstrukturen verlangt, wodurch am Ende die Erweiterungen nur noch von den Framework Architekten selbst bewältigt werden kann [Pree/Koskimies99]. Pree und Koskimies haben auf der Basis der negativen Erfahrungen mit umfangreichen Frameworks einen Vorschlag zum Einsatz der Frameworkprinzipien für kleinere Aufgaben in der Anwendungsentwicklung erarbeitet. Mit so genannten Framelets können für ausgewählte Entwicklungsaufgaben mit hohen Anforderungen an Flexibilität Systembereiche voneinander entkoppelt werden.

Komponentenorientierung

Mit Frameworks wurde die Komposition von Komponenten noch nicht konsequent genug umgesetzt. Schon DeRemer und Kron haben eine Unterscheidung zwischen „Programmieren im Kleinen“ und „Programmieren im Großen“ vorgeschlagen [DeRemer/Kron76]. Shaw und Garlan haben diese Idee noch weiter entwickelt und sie als Architekturprinzip etabliert. Sie formulierten, dass es eine Architektur-Beschreibungssprache geben muss, welche es einem Konstrukteur erlaubt, ein Anwendungssystem hierarchisch in kleinere, beherrschbare Teile zu zerlegen (Komponenten und Verbindungen), um daraus umgekehrt wieder ein Gesamtsystem zu konstruieren [Shaw/Garlan96, 152].

Somit gibt es aus Konstruktionssicht zwei unterschiedliche Architekturebenen:

1. Gesamtsystemebene, wobei Gesamtsysteme aus Komponenten mittels einer „Module Interconnection Language“ [DeRemer/Kron76] zusammengebaut werden.
2. Komponentenebene, die den Technologieeinsatz zur Herstellung der Komponenten beschreibt.

Für die zweite Ebene werden Konstruktionsprinzipien der Objektorientierung und Herstellungsprinzipien wie Frameworks (z.B. Belegframework für Auftragsabwicklung),

Programmgeneratoren und Engines eingesetzt. Engines sind Softwareprogramme mit Multifunktionscharakter, die als Serviceanbieter im Gesamtsystem auftreten (z.B. Pricing-Engine für jegliche Art von Preisberechnung).

Für den Zusammenbau der Komponenten werden einerseits Kommunikationsverfahren und Komponentenverwaltungswerkzeuge (Komponententechnologie) benötigt, andererseits sollten Konstruktionsmethoden für die Komponentenbildung und Komposition eingesetzt werden.

Als Komponententechnologien haben sich für die Entwicklung mit Microsoft-Werkzeugen DCOM (COM+) [Horstmann/Kirtland97, Kirtland97] und .Net remoting [Microsoft02, 106ff] etabliert und für Cross-Plattform-Anwendungen CORBA Components [CCM02]. In der Java Welt war RMI (Remote Method Invocation) der erste Schritt zur Komponententechnologie. Anschließend wurde in Kooperation mit der CORBA Community das IIOP-Protokoll (Internet Inter-ORB Protocol) um die RMI-Funktionalität erweitert [Curtis04], so dass EJB's (Enterprise Java Beans) auf einem J2EE Application Server mit einem CORBA Application Server kommunizieren können. Ein historischer Abriss über komponentenbasierte Softwaresysteme haben Mezini u.a. kompakt vorgelegt [Mezini et.al. 03].

Business-Objekte

Auf der Gesamtsystemebene wurden bei betriebswirtschaftlichen Standard-Anwendungssystemen für die Komponentenbildung Business-Objekte eingeführt. Business-Objekte sind Abbildungen von realen Sachverhalten oder Konzepten aus dezidierten Arbeitsbereichen (Domänen) von Geschäftsanwendungen [BOMSIG94]. Ihnen werden Multi-Sichten auf die Daten und Funktionen der Mikro-Domäne unterstellt. Sie sind in Geschäftsprozess-Ketten eingebunden und stellen eine redundanzfreie Datenhaltung sicher. Sie unterstützen im Idealfall Multi-User, Transaktions-, Security- und Historisierungskonzepte. Mit Business-Objekten lassen sich auf der einen Seite Anwendungssysteme modellieren [Taylor95, Prins96], auf der anderen Seite kann man sie als Integrationsbasis betrachten, die eine komponenten-orientierte Erweiterung von Prozessen und Funktionen in den Anwendungsbereichen ermöglicht [Lang/Kalkmann97]. Für die Anbindung dieser optionalen Erweiterungs-komponenten wurden Schnittstellen der Business-Objekte für Aufrufe unter Verwendung der oben aufgeführten Komponententechnologien bereitgestellt. SAP

Anwendungssysteme haben dafür hervorgehobene Methodenaufrufe als BAPI's (Business Application Programming Interfaces) im so genannten Interface Repository veröffentlicht, dokumentiert und ihre Stabilität für bestimmte Releases zugesichert [Graf97, 72f].

Web Services und serviceorientierte Architektur (SOA)

Eine evolutionäre Weiterentwicklung der Komponententechnologie sind Web Services. Web Services sind nachrichtenbasierte Schnittstellen (Message-Based Interfaces), die über das Internetprotokoll HTTP (Hypertext Transfer Protocol) respektive dem auf HTTP aufsetzenden SOAP (Simple Object Access Protocol) angesprochen werden. Für ihren Aufruf muss eine URI (Uniform Resource Identifier) existieren. Die Interfaces werden mittels einer WSDL (Web Service Description Language) beschrieben und der Inhalt der Message ist ein XML-Dokument (Extensible Markup Language)[11] [He03, Booth et.al. 04].

Web Services wurden ursprünglich für den Datenaustausch zwischen Geschäftspartnern über das Internet entwickelt (B2B – Business to Business). Gleich anschließend erkannte man, dass Web Services auch für die Kommunikation zwischen Anwendungssystemen ideal geeignet sind (A2A – Application to Application). Der nächste Schritt war dann Web Services auch für die innere Struktur von Anwendungssystemen zu verwenden. D.h. jegliche Kommunikation zwischen Systemkomponenten sollte über Web Services organisiert werden, dann spricht man von serviceorientierter Architektur [He03]. Web Services in ihrer ursprünglichen Definition waren nicht geeignet die Kommunikation zwischen den Komponenten betriebswirtschaftlicher Anwendungssysteme transaktionsorientiert, mit Status und Kontext sowie unter Berücksichtigung von Konfiguration zu gewährleisten. Standardisierungsgremien arbeiten weltweit daran diese und andere Aspekte für eine Web Service Kommunikation in der Anwendungssystementwicklung zu definieren. Mit der Standardisierung der Schnittstellen-Technik und des Kommunikationsverhaltens sind jedoch noch nicht die Inhalte (Semantik) der Schnittstellen festgelegt. Eine semantische Festlegung der Web Services für Business-Objekte ist jedoch eine wesentliche Voraussetzung für eine industrielle Fertigung von Anwendungssystemen.

[11] Ausnahme: Binärdaten Anhänge

Business Services-Plattform (BSP)

Eine konsequente Umsetzung der Konzepte von serviceorientierten Architekturen für Standard-Anwendungssysteme bedeutet, dass Werkzeuge für eine serviceorientierte Entwicklung eingesetzt werden müssen (Service-Definition, Service-Verwendung, Service-Komposition). Darüber hinaus sind semantische Definitionen von Services erforderlich. Zudem sollten betriebswirtschaftliche Grundfunktionen, wie beispielsweise Stammdatenverwaltung, belegorientierte Abwicklung, Workflow- und Prozess-management sowie Formular- und Berichtswesen, als Basisfunktionalität durch so genannte **Business Services** bereitgestellt werden. Eine damit neu geschaffene serviceorientierte Entwicklungsplattform kann als Business Services-Plattform bezeichnet werden. Sowohl Microsoft mit ihrem Konzept der Software Factories [Greenfield/Short04] als auch SAP mit ihrem Ansatz der Enterprise Services Architecture [Woods03] gehen in diese Richtung.

Woods definiert am Beispiel der Technologie und den Anwendungssystemen von SAP, wie der Wandel von 3-Schicht-Client/Server-Architekturen hin zu einer vollständig serviceorientierten unternehmensweiten Anwendungssystemlandschaft (Enterprise Services Architecture) aussehen kann [Woods03, 20ff]. Woods beschreibt weiterhin, wie die Kommunikation der Benutzerinteraktionskomponenten (User Interface) mit den jeweiligen SAP-Anwendungen über Web Services realisiert wird. Hinzu kommt, dass mit der Technologie-Plattform SAPNetweaver™ so genannte „Composite Applications“ gebaut werden können. Composite Applications sind Komponenten die aufbauend auf den Web Services der SAP-Anwendungen neue Funktionalität für Benutzergruppen bereitstellen, die von bisherigen SAP-Anwendungen nicht erreicht wurden.

Woods stellt als Plattform-Komponenten der Enterprise Services Architecture nur die Bestandteile der SAPNetweaver™ Integrations- und Applikationsplattform dar. Dies sind [Woods03, 127f]:

- Content Management (Verwaltung unstrukturierter Daten)
- Data Warehouse (Business Intelligence Lösungen)
- Enterprise Application Integration (EAI)
- Portals (Portaltechnik für die Integration und Zusammenführung von Benutzerinteraktionsmöglichkeiten)

- Business Process Management (flexible Prozessmodellierung und nachrichtenbasierte Prozessausführung)
- Collaboration (Kommunikation mit Geschäftspartnern über Web Services)
- Mobile Infrastructure (Technologiebasis für mobile Anwendungen)

Damit lässt Woods das betriebswirtschaftliche Potenzial einer ESA-Architektur der SAP unberücksichtigt. Erst Zencke [Zencke04] erläutert, dass die Enterprise Services Architecture neben den oben angeführten Plattformkomponenten noch

- innovative Konzepte zur Benutzerproduktivitätssteigerung
- ein integriertes analytisches Berichtswesen
- Werkzeuge für Service-Komposition und Composite Applications Entwicklung
- Ausgewählte Business-Objekte, Engines und Anwendungskomponenten als Service-Anbieter

enthält.

Zudem ist die ganze Technologie- und Anwendungslandschaft vollständig in ein Life Cycle Management-Konzept eingebettet. Das soll bedeuten, dass die Werkzeuge und Dienstleistungen der SAP AG für den Betrieb von SAP-Anwendungen in die Enterprise Services Architecture integriert sind. Zencke erläutert, dass alle Services der SAP Anwendungssysteme zukünftig in einem Enterprise Services Repository verwaltet werden.

Noch mehr als bei Business-Objekten werden Business Services-Plattformen als **Integrationsbasis** für eine komponentenorientierte Erweiterung von Anwendungssystemen dienen. Enthalten Business Services-Plattformen neben betriebswirtschaftlichen Grundfunktionen zusätzlich noch zentrale Dienste für Organisationsmanagement, Berechtigungsverwaltung sowie Länder- und Branchenunterstützung, stellen sie eine Basis für mehrstufige Fertigungsverfahren bereit. Durch Konfiguration der Integrationsbasis und durch Entwicklung und Komposition von Composite Applications können dezentrale Entwicklungsabteilungen des Herstellers sowie

Entwicklungspartner und Kunden modifikationsfrei neue Lösungen nach dem Vorbild der industriellen Fertigung entwickeln.

2.2.1 Zusammenfassung

Die technologische Basis für McIlroys Vision [McIlroy69] scheint somit geschaffen. Allerdings fehlt noch eine Konstruktionsmethode, die SOA-basierte Anwendungssysteme ganzheitlich und anwendungsbereichsorientiert für die Herstellung, Anpassung und Variation beschreibt. Standard-Anwendungssysteme müssen eine weitgehende Integration der betriebswirtschaftlichen Belange ihres Anwendungsbereiches sicherstellen und gleichzeitig eine möglichst effektive Entkopplung der Anwendungsprogramme von Technologien zur Präsentation und Ausgabe von Arbeitsergebnissen und für die Kommunikation mit anderen Systemen und Geschäftspartnern realisieren.

Standard-Anwendungssysteme haben eine vergleichsweise hohe Komplexität in den Strukturen ihrer Programmlogik, die dadurch bedingt ist, dass ihr Einsatz in vielen verschiedenen Kundenszenarien nur durch eine hohe Variabilität in der betriebswirtschaftlichen prozessualen und funktionalen Abwicklung erreicht werden kann. Eine Verbesserung der inneren Systemarchitektur von Anwendungssystemen wurde durch Datenmodellierung, Objektorientierung, Frameworkkonzepte und Business-Objekten erreicht. Das bedeutet mehr Stabilität in den Fachkonzepten durch eine verbesserte Orientierung am Anwendungsbereich. Erst standardisierte Komponententechnologien mit ihrer Weiterentwicklung hin zu Web Services und serviceorientierter Architektur und noch einen Schritt weiter hin zur Business Services-Plattform haben flexible Herstellungsverfahren mit mehrstufiger Fertigung und komponentenorientierter Anpassung und Erweiterung möglich gemacht.

Für das „Programmieren im Großen“ respektive für eine Architektur-Beschreibungssprache wurde noch keine Konstruktionsmethode entwickelt, die das Entwickeln von Baukastensystemen, ein Konstruieren mit Lösungskatalogen und Variantenkonstruktion für Branchen- und Kundenlösungen unterstützt. Eine Konstruktionsmethode, die diese Prinzipien auf die Konstruktion und Herstellung von betriebswirtschaftlichen Standard-Anwendungssystemen anwendet, muss sich an den Aufgaben des Anwendungsbereiches orientieren und sie muss sich auf die Fachsprache des Anwendungsbereiches abstützen.

2.3 Sprachbasierte Anwendungsentwicklung

In technischen Systemen sind alle Bauteile konstruiert, d.h. es werden nicht zufällig entworfene Maschinenelemente zu Gesamtsystemen zusammengebaut. Auch in der Softwareentwicklung sind konstruktive Ansätze zu finden. Beispielsweise werden Grafikprogramme nicht entwickelt, indem Zeichenelemente empirisch definiert werden. Kreise, Ellipsen, Quadrate sind nicht „in der Regel“ so wie sie sind, sondern sie werden mathematisch exakt definiert (konstruiert).

Mit demselben Anspruch müssen betriebswirtschaftliche Anwendungssysteme entwickelt werden. Bauteile für Anwendungssysteme, wie Datenelemente, Objekte, Methoden, Web Services, Oberflächentexte, Kennzahlen und andere, müssen in ihrer Semantik sowohl auf Konstruktionsebene als auch auf Programmebene exakt definiert (rekonstruiert) werden [Ortner et.al. 99, Schienmann97, ebXML04].

Nicht nur die Bauteile, auch das Verhalten von Anwendungssystemen, d.h. Funktionen, Prozessen und Benutzerinteraktionen, müssen mit einer terminologisch und grammatisch normierten Fachsprache beschrieben (Fachentwurf) und unter Verwendung derselben entwickelt werden (Implementierung). Ein Anwendungssystem wird immer für einen bestimmten betriebswirtschaftlichen Anwendungsbereich entwickelt. Demzufolge ist für diesen Anwendungsbereich die Sprache zu rekonstruieren und zu normieren. Eine Sprache besteht im Wesentlichen aus einem Wortschatz, der in einem Lexikon hinterlegt wird, aus einer Grammatik in Form von Satzbauplänen und einer kategorialen Gegenstandseinteilung. Ortner hat die Eigenschaften einer Normsprache folgendermaßen festgelegt [vgl. Ortner05, 263]:

Gegenstandseinteilung:	Dinge, Geschehnisse, Eigenschaften und Beziehungen;
Grammatik:	Hauptaussagen und Zusatzaussagen;
Terminologie:	Objekt- und Metasprache.

Wichtig ist, dass die Gegenstandseinteilung nicht methodenspezifisch für einen Entwicklungsansatz durchgeführt wird. Zuerst sollte eine methodenneutrale Gegenstandseinteilung erstellt werden, die dann eine Grundlage für methodenspezifische Fachkonzepte bildet.

Für den Aufbau einer normierten Fachsprache (Normsprache) für einen bestimmten Anwendungsbereich müssen Aussagen und Begriffsdefinitionen **dialogisch,** beispiels-

weise durch Interviews, **hermeneutisch,** beispielsweise durch Fachliteratur und **praktisch,** beispielsweise durch Beobachtung, Mitarbeit und Experiment zusammengetragen werden [Ortner05, 55]. Die Aussagen liegen damit in einer Gebrauchssprache vor. Der Rekonstruktionsvorgang beginnt mit der Normierung der Begriffe. Sie müssen von Mehrdeutigkeiten, Unschärfen, Widersprüchen und falscher Bedeutungszuordnung befreit werden [Ortner/Söllner89, Schienmann97, Lehmann99]. Die normierten Fachbegriffe werden dann alphabetisch sortiert, mit expliziten Definitionen, Beispielen und Gegenbeispielen, Einsatzbeispielen und Beziehungen in einem Lexikon abgelegt [Ortner97, 97].

Nach den Fachbegriffen wird die Grammatik normiert, „so dass von der normierten Aussage eindeutig auf den dargestellten Sachverhalt geschlossen werden kann“ [Schienmann97, 147]. Schienmann hat eine Übersicht geeigneter Satzbaupläne für die Einordnung und Reduktion der gebrauchssprachlichen Aussagen – jetzt mit normierten Fachbegriffen – zusammengestellt [Schienmann97, 164ff].

Satzbaupläne sind Schablonen für eine rationelle Normierung der grammatischen Strukturen einer Aussagensammlung. Die grammatischen Kategorien der Aussagen lassen sich auch hierarchisch in einem Strukturbaum darstellen. Dabei werden Sätze in Konstituenten[12] zerlegt und gemäß ihrer Dominanz und Reihenfolge in einer Hierarchie abgebildet [Bußmann90, 745f]. Wird diese Form der grammatischen Analyse weiterentwickelt, kann damit eine Wort- und Satzteilverwendung aussagenübergreifend analysiert werden. Es ist möglich, für einen Aussagenbestand bzw. für einen betrachteten Anwendungsbereich die Einsatzmöglichkeiten von Worten und Satzteilen innerhalb der Vielfalt aller Sätze in eine strukturelle und mengenorientierte Darstellung zu überführen.

In der Produktionswirtschaft werden Stücklisten für die Verwaltung von Erzeugnisstrukturen für Produkte eingesetzt. Damit werden die Beziehungen und Mengenverhältnisse von Materialien verwaltet, die in die Herstellung eines Produktes einfließen [VDI 2215-80, 8ff, Loos99, 7ff, Ortner05, 143ff]. Dieses Prinzip einer mengenorientierten Analyse von Erzeugnisstrukturen lässt sich auch auf eine mengenorientierte Analyse der Wortverwendung in Aussagensammlungen anwenden. Becker u.a.

[12] Eine Konstituente ist „jede sprachliche Einheit (Morphem, Wort oder Syntagma), die Teil einer größeren sprachlichen Einheit ist“ [Bußmann90, 413].

[Becker et.al. 03] haben dafür das Verfahren der Varianten-Stücklisten von Wedekind und Müller [Wedekind/Müller81] vorgeschlagen. In graphischer Darstellung von Varianten-Stücklisten werden Worte, Satzteile und Sätze als Knoten der Stücklisten dargestellt. Die Verbindungskanten zwischen den Knoten stellen mit Mengenangaben versehen eine Beziehung zu einem oder mehreren übergeordneten Knoten dar. Die Beziehungen sind je nach Knotentyp unterschiedlich ausgeprägt. Werden die Knotentypen von Wedekind und Müller für die Aussagenanalyse zu Grunde gelegt, so gibt es Teileknoten (Worte und elementare Satzbauteile), Konjunktivknoten (Sätze und zusammengesetzte Satzbauteile), Alternativknoten (alternative Worte oder Satzbauteile) und so genannte leere Knoten, die es ermöglichen eine Auswahl als optional zu betrachten. In offenen Varianten-Stücklisten werden Varianten innerhalb von Varianten verwaltet, indem der größtmögliche Funktionsumfang eines Erzeugnistyps einschließlich aller Varianten gespeichert wird [Grupp95]. Damit lassen sich alle zulässigen Einsatzmöglichkeiten von Worten und Satzbauteilen abbilden. Stücklisten sind das Ergebnis einer Analyse des Aufbaus von komplexen Aussagen. Werden Stücklisten synthetisch aufgelöst, so ist das Ergebnis ein Teileverwendungsnachweis [Ortner05, 144]. Ein Teileverwendungsnachweis liefert Informationen über den tatsächlichen Einsatz von Fachbegriffen und Satzbauteilen im betrachteten Anwendungsbereich.

Liegen die anwendungsbereichsspezifischen Aussagen mit rekonstruierten Fachbegriffen in normierten Satzbauplänen vor und sind zudem die möglichen Verwendungsweisen von Fachbegriffen in Aussagen mittels Varianten-Stücklisten definiert, so ist eine Normsprache für den Fachentwurf erstellt. Anschließend kann mit nachfolgendem Erfassungsschema eine kategoriale Gegenstandseinteilung durchgeführt werden.

Gegen-stand	**Aufbau**		**Ablauf**
	dingorientiert	geschehnisorientiert	
nach innen			
	Einschränkungen		
nach außen			

Tabelle 2: Schema zur Erfassung entwicklungsrelevanter Aussagen im Fachentwurf [Ortner05, 64]

Mit Ortners Erfassungsschema [Ortner05, 63] kann die Aussagensammlung für einen Gegenstand auf Vollständigkeit überprüft werden – Alle Aspekte müssen vertreten sein. Die Kategorien „nach innen“ entsprechen den Eigenschaften, die Kategorien „nach außen“ den Beziehungen der Gegenstände. Sind die Aussagen noch unvollständig, so können unter Verwendung der Normsprache weitere Aussagen formuliert (konstruiert) werden. Die neu erstellten Aussagen werden mit Fachexperten des Anwendungsbereiches geprüft und sind damit Teil des methodenneutralen Fachentwurfs. Die kategoriale Gegenstandseinteilung nimmt eine zentrale Rolle ein, für die Klassifikation und Transformation eines methodenneutralen Fachentwurfs in einen methodenspezifischen Systementwurf. Denn gerade die Gegenstandseinteilung ist die Modellierungsbasis eines Architekturkonzeptes. Eine Übersicht der Übergänge zwischen Entwurfs- und Realisierungsphasen für eine sprachbasierte Anwendungsentwicklung zeigt das Multipfad-Vorgehensmodell von Ortner.

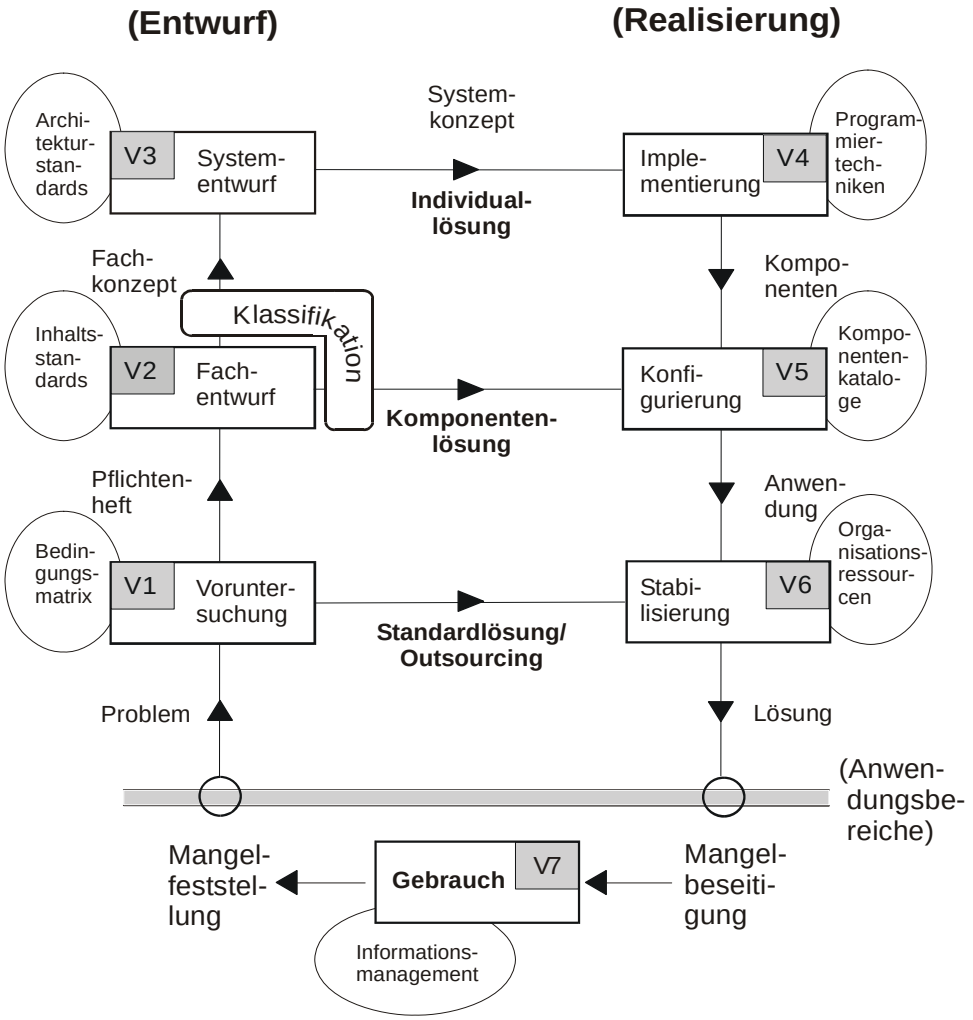


Abbildung 14: Multipfad-Vorgehensmodell für die Anwendungssystem-Entwicklung [Ortner05, 39]

Es kann erst wirklich von einer sprachbasierten Anwendungsentwicklung gesprochen werden, wenn alle Phasen einer Anwendungsentwicklung auf derselben normierten Fachsprache basieren bzw. ihre jeweilige Sprache nach genau festgelegten Regeln davon abgeleitet oder darauf abgebildet werden kann.

Ortners Vorgehensmodell stellt die möglichen Pfade für die Neukonstruktion und Weiterentwicklung von Anwendungssystemen dar. Wird der Fokus auf die Phasen V1, V2, V5 und V6 gelegt, so kann damit ideal eine katalogbasierte Variantenkonstruktion mit Baukastensystemen abgebildet werden. Mit V1, der Voruntersuchung, findet dann die Anforderungsanalyse der Einsatzmöglichkeiten eines Standard-Anwendungssystems beim Kunden statt. Im Fachentwurf (V2) findet ein Abgleich der Aussagen (Anforderungen) aus der Voruntersuchung mit den bereits vorhandenen Lösungen statt. Ist eine passende Lösung vorhanden, dann besteht die Konstruktionstätigkeit nur noch darin, die Komponenten für diese Lösung auszuwählen und zu konfigurieren. Andernfalls ist durch Rekonstruktion und Normierung der Anforderungen ein Fachentwurf für neue Lösungsvarianten für das Baukastensystem zu erstellen. Daran anschließend kann dieser Entwurf weiter konkretisiert werden, indem er in ein Komponentenkonzept für den Systementwurf (V3) überführt (transformiert) wird. Dabei werden die Aussagen der Gegenstandseinteilung für das Design der Granularität und des Schnittstellenangebotes der neuen Komponenten zu Grunde gelegt. In Phase V4 werden die Komponenten implementiert und stehen damit dem Baukastensystem für die Konfigurierung (V5) zur Verfügung. Die Gesamtlösung bzw. Ergänzungslösung wird in der Phase V6 beim Unternehmen des Kunden eingeführt.

Durch Einsatz (V7) und Mangelfeststellung findet eine beständige Verbesserung ihren Weg in die nächste Voruntersuchung. Mit dem Multipfad-Vorgehensmodell wird der so genannte „Customer Life Cycle“ vollständig mit konstruktivem Einsatz der Fachsprache des Anwendungsbereichs und den entsprechenden Transformationen in methodenspezifische Entwicklungsresultate abgedeckt.

2.4 Diskussion

Ziel der vorliegenden Arbeit ist es, Ansätze zu entwickeln, wie man den erkennbaren Mangel an systematischer Konstruktionsmethodik in der Anwendungssystementwicklung beheben könnte. Der zu Grunde liegende Forschungsauftrag war, die Konstruktionslehre klassischer Ingenieurbereiche zu analysieren und ihre Methoden

und Verfahren auf die, im Vergleich junge Konstruktionstheorie für betriebswirtschaftliche Anwendungssysteme zu übertragen.

Schon das Vorgehen zur Untersuchung dieser Aufgabe entspricht konsequenterweise einem Vorgehensmodell für technische Entwicklung und Konstruktion (vergleiche dazu Abschnitt 2.1.1). In einem Meta-Vorgehensmodell[13] zur Entwicklung einer Konstruktionsmethode werden die folgenden Arbeitsschritte durchlaufen:

1. **Klären der Aufgabenstellung**
   In der Einleitung und in der Einführung in Kapitel 2 werden Zielsetzung, Problemstellung und Untersuchungsaspekte der Gesamtaufgabe ausführlich dargestellt.

2. **Ermitteln der Funktionsstrukturen – Methodenanalyse der technischen Konstruktionslehre**
   In diesem Arbeitsschritt werden die Grundlagen der technischen Konstruktionslehre in einem Gesamtüberblick dargestellt und untersucht. Damit wird ein Verständnis für ein methodisches Vorgehen im Sinne der technischen Konstruktionslehre geschaffen. Das Ergebnis daraus dient der weitergehenden Ermittlung von Funktionsstrukturen für eine Konstruktionsmethode von betriebswirtschaftlichen Standard-Anwendungssystemen.

3. **Suchen nach methodischen Lösungsprinzipien**
   Wichtige Verfahren und methodische Hilfsmittel werden als Lösungsprinzipien für technische Konstruktionsmethoden folgendermaßen zusammengefasst:
   - Methodisches Vorgehen
   - Ansätze zur Problemlösung
   - Systematische Lösungsentwicklung
   - Baukastensysteme
   - Sachmerkmalleisten

   Eine Betrachtung der verschiedenen Konstruktionsarten vervoll-

[13] Ein Meta-Vorgehensmodell ist ein Vorgehensmodell zur Entwicklung einer Methode (planmäßiges Vorgehen).

ständigt den Überblick der methodischen Lösungsprinzipien technischer Konstruktionslehre.

4. **Iterativer Rücksprung zu Schritt 2 und Ergänzung um eine Methodenanalyse der Konstruktionstheorie für Anwendungssysteme**
   Für eine Prüfung, ob die Lösungsprinzipien der technischen Konstruktionslehre auf die Anwendungssystementwicklung angewendet werden können, muss zuerst eine Grundlagenanalyse der Rahmenbedingungen von Konstruktionsmethoden für Anwendungssysteme durchgeführt werden. In diesem Arbeitsschritt werden in einem historischen Überblick die konstruktiven Architekturansätze von Anwendungssystemen untersucht und dargestellt (siehe Abschnitt 2.2).

5. **Anwendung der Lösungsprinzipien aus Schritt 3 auf Anwendungssystem-Architekturen**
   Die Untersuchung zeigt, dass mit einer serviceorientierten Architektur und, noch weitergehend, mit einer Business Services-Plattform die technologischen Grundlagen für industrielle Fertigungsverfahren von Anwendungssystemlösungen gegeben sind. Allerdings können die in Schritt 3 gefundenen methodischen Lösungsprinzipien erst dann auf die Konstruktion von Anwendungssystemen angewandt werden, wenn eine stärkere Anwendungsbereichsorientierung erfolgt. Dazu muss methodisch eine semantische Rekonstruktion der Beschreibungssprache von Anforderungs-, Entwurfs- und Entwicklungsergebnissen durchgeführt werden.

6. **Iterativer Rücksprung zu Schritt 2 und Ergänzung um eine Methodenanalyse der sprachbasierten Anwendungsentwicklung**
   In Abschnitt 2.3 werden methodische Verfahren zum Aufbau einer normierten Fachsprache (Normsprache) mit Gegenstandseinteilung, Grammatik und Terminologie vorgestellt. Die Verwaltung der Normsprache wird in Form normierter Aussagen, eines Lexikons und Varianten-Stücklisten für zulässige Einsatzmöglichkeiten der Termini und Satzbauteile durchgeführt.

7. **Suche nach methodischen Lösungsprinzipien der sprachbasierten Anwendungssystementwicklung**
   Es wird ein methodenneutraler Fachentwurf, der auf einer Normsprache basiert, vorgeschlagen. Ihre Gegenstandseinteilung ist ein wesentlicher Erfolgsfaktor für eine methodenspezifische Klassifikation der Aussagen, damit der Übergang in einen Systementwurf oder eine Auswahl von bereits fertiggestellten Komponenten effizient durchgeführt werden kann. Für eine ganzheitliche Betrachtung der Lösungsprinzipien einer sprachbasierten Anwendungsentwicklung werden alle Phasen zur Erstellung eines Gesamtsystems in ein Multipfad-Vorgehensmodell eingebettet und bilden damit den methodischen Unterbau für die Konstruktion von Anwendungssystemen.

8. **Gliedern in realisierbare Module (Bausteine) der Konstruktionsmethode**
   Als Ergebnis der vorangegangenen Untersuchungen können folgende Methodenbausteine identifiziert werden:
   - Gestaltung der Komponenten und ihres Baukastensystems
   - Konstruktionsprinzipien und Einflussfaktoren einer Variantenkonstruktion
   - Konstruktion mit Lösungskatalogen: Gliederung und Zugriffsteil in Form eines Ordnungssystems und Hauptteil in Form eines Baukastensystems

Weitere Arbeitsschritte des Meta-Vorgehensmodells sind das **Gestalten der maßgebenden Methodenbausteine** und das **Gestalten der Gesamtmethode**. Im weiteren Verlauf der Arbeit werden auf der Grundlage von Kapitel 2 die Methodenbausteine ausgearbeitet (gestaltet) und zu einer Gesamtmethode zusammengeführt. Die Gesamtmethode wird aufgrund ihrer Anwendungsbereichsorientierung und ihren ausdrücklich betriebswirtschaftlich-inhaltlichen Designprinzipien für Komponenten und Konstruktionshandlungen (Variantenkonstruktion durch Auswahl) als ***Methode der semantischen Komposition*** bezeichnet. Der letzte Arbeitsschritt des Meta-Vorgehensmodells ist das **Ausarbeiten der Nutzungsangaben** für die ***Methode der semantischen Komposition***. In Kapitel 4 werden anhand eines Beispiels und unter Verwendung eines dafür entwickelten Werkzeugs die Ergebnisse der vorliegenden Arbeit dokumentiert.

Kapitel 3 Aufbau und Inhalt der Methode der semantischen Komposition von Anwendungssystemen

In der vorhergehenden Diskussion hat sich gezeigt, dass die Anforderungen einer individualisierbaren Anwendungssystemlösung aus Standardbauteilen sehr gut durch eine Kombination der Variantenkonstruktion und Katalogkonstruktion basierend auf einem Baukastensystem erfüllt werden könnten. Die Herausforderung besteht nun darin, eine geeignete Form für die „Standardbauteile“ zu finden. Vergleichbar der technischen Konstruktionslehre muss ein geeigneter, ganzheitlicher Methodenkontext gezielt den Anforderungen der Anwendungssystementwicklung gerecht werden.

Mit der ***Methode der semantischen Komposition*** wird hier ein Ansatz vorgestellt, der einen Rahmen erstellt, wie die Bauteile von Standard-Anwendungssystemen gestaltet werden können und wie der Vorgang der konstruktiven Konfiguration von vorhandenen Lösungen in einen **sprachbasierten Methodenkontext** eingebettet wird. Die nachfolgenden Ausführungen beschreiben den Aufbau und den Inhalt der ***Methode der semantischen Komposition*** durch Betrachtung vielfältiger Problemstellungen und ihrer möglichen Lösungsalternativen. Folgende Fragestellungen sind dabei zu untersuchen:

- Welche **Arten** von Komponenten gibt es?
- Wie sieht die Elementarisierung des **Baukastensystems** aus?
- Wie kann eine **Arbeitsteilung** im Entwicklungs- und Konstruktionsprozess aussehen?
- Wie kann das **Ordnungssystem** für die Bauteile aufgebaut werden?

Im folgenden Abschnitt wird zuerst eine Unterteilung der Komponenten in der Anwendungssystementwicklung vorgenommen. Anschließend werden die zwei aneinandergrenzenden Schichten der Komponenten ausführlich in Bezug auf die Entwicklung von Anwendungssystemen betrachtet. Ihre gegenseitige Beeinflussung wird dabei verdeutlicht. Ein Beispiel mit Anwendungselementen begleitet die weiterführenden Lösungsansätze. Der Aufbau eines Ordnungssystems für Anwendungselemente schließt dieses Kapitel ab.

3.1 Unterteilung der Komponenten nach Abstraktionsebenen

Für eine Gliederung der Komponenten von Anwendungssystemen entsprechend ihrer Abstraktionsebenen wird hier eine Einteilung in

- Anwendungsbereichsebene und
- Herstellungsebene

vorgeschlagen [Ortner et.al. 99]. Wobei die Herstellungsebene wiederum differenziert als

- Montageebene und
- Ebene der Komponentenproduktion

betrachtet werden kann.

Die Einteilung in Anwendungsbereich und Herstellung entspricht der im Maschinenbau geläufigen Differenzierung zwischen Konstruktionselementen und Maschinenelementen [Steinhilper/Röper94, 1]. „Konstruktionselemente“ für betriebswirtschaftliche Anwendungssysteme sind demnach elementare Aufgabenbausteine, die in Bezug auf die Aufgabe nicht mehr weiter zerlegbar sind. Die Definition von Steinhilper und Röper muss auf Baugruppenstrukturen erweitert werden, denn sowohl im Maschinenbau als auch in der Anwendungssystementwicklung wird mit Kompositionen (Baugruppen) als Konstruktionselementen konstruiert. Diese Baugruppen entsprechen dann entweder umfangreicheren Aufgaben oder sie stellen bereits gruppierte Aufgabenkomplexe dar, die aber wie ein elementares Konstruktionselement im Konstruktionsprozess verwendet werden[14].

Konstruktionselemente werden hier in Abgrenzung zu anderen Komponenten der Anwendungssystementwicklung als „Anwendungselemente“ bezeichnet.

Definition:

> **Anwendungselemente** sind Komponenten, die sich auf der Abstraktionsebene des (potentiellen) Anwenders zur Erfüllung von funktionalen und prozessualen Aufgaben in seinem Tätigkeitsbereich direkt identifizieren lassen.

Anwendungselemente stellen eine Teilmenge aller Komponenten der Anwendungssystementwicklung dar. Sie besitzen Eigenschaften, die für die Verwendung in einem Konstruktionsprozess, der durch Auswahl und Komposition charakterisiert ist, sehr nützlich sind. Dieser Konstruktionsprozess wird im Folgenden vorgestellt.

Anwendungselemente sind **semantische Denkeinheiten der Anwender**, die ihr Arbeitsumfeld beschreiben. Ein Anwendungselement besteht aus fachlicher Konzeption und Implementierungsvorschrift[15]. In wenig ausgereiften Baukastensystemen besitzt ein Anwendungselement nicht unbedingt eine Implementierungsvorschrift. Es

---

[14] In besonderen Fällen entstehen durch die Komposition von Elementarbausteinen Multigegenstände. Das sind dann Konstruktionselemente, die ursprünglich alternative Eigenschaften in sich vereinigen (siehe Abschnitt 3.4.2).

[15] Eine Implementierungsvorschrift ist eine Systeminstruktion, die entweder ein Programm aktiviert, ein Programm verteilt oder eine Konfiguration aktiviert bzw. eine Konfiguration verteilt. In Abschnitt 3.2.1 wird dieser Zusammenhang noch im Detail erläutert.

besteht dann nur aus einem Konzept. Ein Anwendungselement als Konzept (mit anderen Worten eine abstrakte Komponente) kann im Konstruktionsprozess sehr wohl seine Funktion erfüllen und stellt anschließend eine Anforderung für die Herstellung der Implementierungsvorschrift des Anwendungselementes dar. Beschreibende Informationen (Eigenschaften, Beziehungen, Dokumentation, Design, Schnittstellenbeschreibungen, Einordnung in übergreifende Konzepte) sind ebenfalls Bestandteile der Anwendungselemente.

„Maschinenelemente“ werden im Bereich der Anwendungssystementwicklung als Bausteine zur Herstellung von Konstruktionselementen gesehen. Die den Maschinenelementen vergleichbaren Komponenten der Anwendungssystementwicklung werden hier „Creatorelemente“ genannt.

Definition:

> **Creatorelemente** sind Komponenten, die sich auf der Abstraktionsebene des Softwareentwicklers oder Contententwicklers für die Herstellung von Anwendungselementen identifizieren lassen.

Auf mögliche Realisierungsformen von Creatorelementen wird in den folgenden Abschnitten näher eingegangen.

Für eine konstruktive, komponentenorientierte Standardsoftwareentwicklung wird in dieser Arbeit die Entwicklung eines Baukastensystems vorgeschlagen (siehe Abschnitt 3.3.7). Es ist jedoch sinnvoll, für die Anwendungsbereichsebene und für die Herstellungsebene zwei verschiedene Baukastensysteme zu entwickeln. Denn die Elementarisierung der Gesamtsystemvarianten (Produktfamilie) in ihre Bestandteile unterliegt jeweils einer anderen Absicht. Ein Baukastensystem für die Anwendungsbereichsebene, ist als externes Baukastensystem in Richtung Partner oder Kunde zu betrachten. Dadurch wird eine flexible und schnelle Lösungsentwicklung zur Erfüllung von Marktsegmentanforderungen (Branchenkonfigurationen) oder von Kundenanforderungen möglich (Kundenkonfigurationen).

Creatorelemente basieren im Gegensatz zu den Anwendungselementen auf der systemorientierten Sichtweise des Software- oder Contententwicklers und benötigen somit ein eigenes, auf ihre Anforderungen zugeschnittenes Ordnungssystem. Dabei stehen Versionsverwaltung, Wiederverwendung und Technologiebezug im Vorder-

grund. Für die Verwaltung von Anwendungselementen wird ein Ordnungssystem benötigt, welches sich auf den Anwendungsbereich des zugrunde liegenden Baukastensystems ausrichtet (siehe Abschnitt 3.4).

Die nachfolgende Grafik zeigt eine Übersicht der einzelnen Bereiche einer komponentenorientierten Anwendungssystementwicklung. Für eine terminologische Integration von Ordnungssystem I (für Anwendungselemente) mit Ordnungssystem II (für Creatorelemente) wird eine einheitliche normierte Fachsprache (Normsprache) verwendet. Diese Normsprache [Ortner97], determiniert durch die Anwendungsbereiche, stellt eine sprachlich eindeutige Beschreibung der Zielsysteme (Kundenanforderungen) sicher [Ortner et.al. 99].

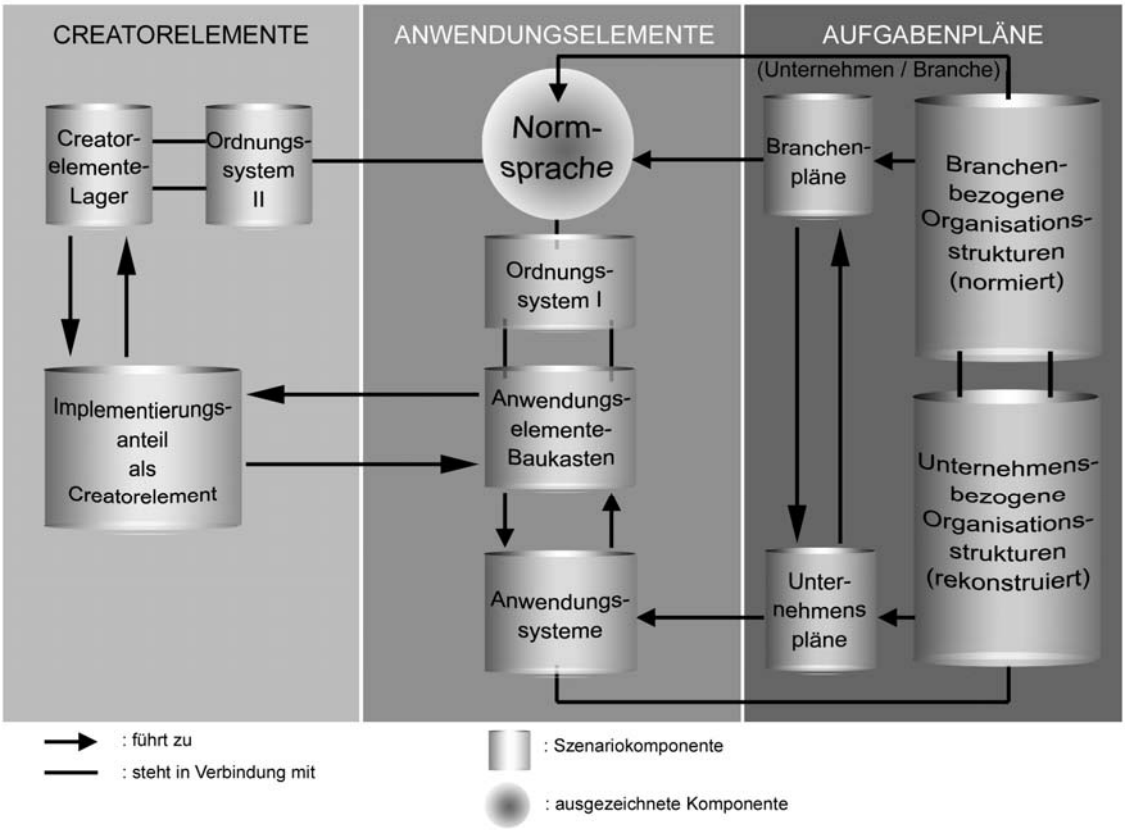


Abbildung 15: Szenario für die komponentenorientierte Anwendungsentwicklung

In Abschnitt 3.4.6 wird die Rolle der Normsprache in der Anwendungssystementwicklung weiter vertieft. Für die weitere Erläuterung von Aufgabenplänen wird in Abschnitt 4.2 ein Beispiel dargestellt.

3.2 Komponentenorientierung auf Herstellungsebene

Die Vorteile der Entwicklung von Anwendungssystemen aus Einzelteilen (Komponenten), die je eine Aufgabe innerhalb der Systeme zuverlässig erfüllen sollen, lassen sich durch die Begriffe „Arbeitsteilung“, „Wiederverwendung“, „Zuverlässigkeit“, „Beherrschbarkeit“ und „systematische Variation“ benennen. Diesen Vorteilen stehen jedoch die Nachteile einer Zerlegung von Systemen gegenüber (z.B. „Schnittstellenprobleme“ und „die Frage nach der Sicherstellung von Gesamtfunktionen“). Ein Einsatz von Methoden technischer Konstruktionsbereiche hilft, die genannten Nachteile bei gleichzeitiger Erzielung von Vorteilen zu reduzieren.

Der Duden definiert Komponenten folgendermaßen:

Definition:

> **Komponenten** [lat.], (Bestand)teile, aus denen sich ein Ganzes zusammensetzt oder in die es zerlegt werden kann [Duden96].

Diese sehr weit gefasste Festlegung des Begriffs lässt aus Sicht der Anwendungsentwicklung jegliches identifizierbare Teil eines Ganzen für die Bezeichnung „Komponente“ zu. Es ist dabei unerheblich, welcher Abstraktionsebene dieses Teil angehört. Die Abstraktionsebenen lassen sich von einzelnen Bits (jedes Programm führt in letzter Konsequenz zu einem Bitstrom [Pree97, 17f, Hansen et.al. 92, 12f]) über Speicherroutinen, Betriebssystem-APIs, Programmfunktionen, Dialoge, Business-Objekte bis hin zu kompletten Anwendungssystemen (z.B. ein komplettes Warenwirtschaftssystem als Teil einer Anwendungsarchitektur) spannen. Der Komponenten-Begriff wird für die Verwendung in der ***Methode der semantischen Komposition*** in den folgenden Abschnitten präzisiert.

3.2.1 Herstellung von Anwendungselementen

In der vorliegenden Arbeit werden alle Bestandteile eines Standard-Anwendungssystems als Komponenten bezeichnet, die mittelbar oder unmittelbar zur Herstellung von Konstruktionselementen verwendet werden. Konstruktionselemente, welche die Aufgabensicht eines Anwendungsbereichs repräsentieren, sind bereits als Anwendungselemente eingeführt.

Jede als Komponente identifizierbare Einheit, die auf der Anwendungsbereichsebene zur Aufgabenerfüllung identifiziert wird, kann als Anwendungselement bezeichnet werden. Beispiele für ein ESA-basiertes Standard-Anwendungssystem sind folgende Einheiten:

- **Softwarekomponenten und Composite Applications**
  Mit Komponententechnolgie und Web Services hergestellte Einheiten, die über standardisierte Schnittstellen mit den Business-Objekten der Integrationsbasis kommunizieren bzw. durch Konfiguration aktivierbare Teile von Business-Objekten darstellen.

- **Konfigurations-Parameter-Pakete**
  Die Variabilität von Standard-Anwendungssystemen wird über einstellbare Konfigurationsparameter repräsentiert. Konkrete Einstellungen zusammengefasst zu Paketen, die genau einer (Teil-) Aufgabenvariante entsprechen, stellen ebenfalls ein Anwendungselement dar.

- **Portal-Pakete und Konfiguration für Benutzerschnittstellen**
  Portalinhalte werden zu Portal-Paketen zusammengefasst und an Kunden ausgeliefert. Benutzerschnittstellen (User Interfaces) für die Bearbeitung von Aufgaben werden über Metadatenkonfiguration in ihrem Verhalten festgelegt und an Business Services gebunden. Auch diese Konfiguration muss an Kunden ausgeliefert werden.

- **Formulare und Berichte**
  Für das Erstellen von Formularen und Berichten werden Werkzeuge bereitgestellt, die durch Deklaration der Eigenschaften für Inhalt und Darstellung die gewünschten Formulare und Berichte als betriebswirtschaftlich identifizierbare Einheit erstellen.

- **Business to Business - B2B-Schnittstellen-Definitionen**
  Standards für die systemgestützte Kommunikation mit Geschäftspartnern werden als Beschreibung der Schnittstellen Definition (Interface-Definition) für die Enterprise Application Integration-Lösung des Standard-Anwendungssystems erstellt.

Anwendungselemente bestehen nicht aus diesen Komponenten, sondern sie enthalten eine Implementierungsvorschrift für diese Komponenten, damit die Aktivierung der Systemrepräsentanz für das Anwendungselement in einem installierten Anwendungssystem durchgeführt werden kann. D.h. eine Composite Application wird durch das Ausführen der Implementierungsvorschrift auf dem System installiert (deployed) und aktiviert. Dasselbe gilt für die anderen oben angeführten Beispiele.

Anwendungselemente müssen so hergestellt werden, dass sie auf der Anwendungsbereichsebene für die Konstruktionshandlung „Komposition“ direkt verwendet werden können. Das ist der Unterschied zwischen technischen Produkten mit physischer Existenz und Softwareprodukten, deren Existenz sich in Form von Vorschriften für Computerprozessoren und Speichermedien ausdrückt. Denn die Komposition von Anwendungselementen wird durchgeführt als Konfiguration und Aktivierung der Implementierungsvorschriften für Anwendungssystem-Komponenten und entspricht dadurch der Montage von Bauteilen und Baugruppen in der industriellen Fertigung.

Die Granularität von Anwendungselementen sowie ihre Beschreibungsform und ihr Einbinden in die Konstruktionsmethode auf Anwendungsbereichsebene wird in Abschnitt 3.3 gezeigt.

3.2.2 Herstellung von Creatorelementen

Die Einheiten, auf die eine Implementierungsvorschrift der Anwendungselemente verweist, wurden als Creatorelemente eingeführt. Zur Herstellung von Creatorelementen sind alle Techniken, die in einem Standard-Anwendungssystem für die Herstellung von den oben aufgeführten Komponenten bereitgestellt werden, einsetzbar. Die Herstellung erfolgt nicht notwendigerweise als Implementierung von neuem Programmcode, sie kann auch durch Konfiguration, Deklaration und Metadatenmodellierung (Contententwicklung) erfolgen. Die Grenze zwischen Softwareentwicklung und Contententwicklung ist nicht mehr eindeutig definiert, denn Contententwicklung wurde in der Vergangenheit als nachgelagerter Fertigungsschritt gesehen. Im Rahmen einer BSP-basierten Anwendungssystementwicklung werden jedoch schon die ersten Entwicklungsergebnisse durch Contententwicklung hergestellt.

Nachdem das Verständnis von Komponenten in der Standard-Anwendungssystem-entwicklung an die Realität angepasst wurde und dadurch neben Softwarekomponenten noch andere Entwicklungsergebnisse beinhaltet, muss auch die Verwendung von Schnittstellen und Komponentenkommunikation angepasst werden. Der Begriff „Schnittstelle“ auf der Entwurfsebene ist nicht an eine Implementierung der Schnittstelle gebunden. Deshalb sollte er auch nicht notwendigerweise mit dem Schnittstellenverständnis der Komponententechnologien (DCOM, CORBA, Web Services) gleichgesetzt werden (Call-Interface). Das Konzept der „Module Interconnection Languages“ (MIL) sieht vor, dass Schnittstellen (Ressourcen) auch für nicht-codierte Komponenten definiert sein müssen [DeRemer/Kron76, Prieto-Diáz/Neighbors86].

D.h. eine Konstruktionsmethode für „Programmieren im Großen“ verwendet die Schnittstellendefinition auf der Entwurfsebene für die Abbildung von Beziehungen zwischen den Komponenten, auch wenn diese nicht durch einen Aufruf eines konkreten Interfaces zur Laufzeit des Anwendungsprogrammes realisiert ist. Abhängigkeiten zwischen Komponenten können durch tief liegende betriebswirtschaftliche Zusammenhänge bedingt sein. Beispielsweise kann ein User Interface für eine Auftragsposition eine Konfiguration für die Darstellung einer Seriennummer enthalten. Diese Darstellung sollte dynamisch aktiviert werden, je nachdem wie die Konfiguration der Artikeleigenschaft „Seriennummer“, des in der Auftragsposition verwendeten Artikels eingestellt ist.

Es kommt im Grunde nicht darauf an, mit welcher Technologie Anwendungssystem-Komponenten realisiert werden. Es ist nur wichtig, dass die variablen Teile eines Standard-Anwendungssystems nach dem Baukastenprinzip kombinierbar sein müssen und, dass das Wissen des Anwendungsbereiches einer ganzen Branche[16] sowie konkrete Ausprägungen für Kunden-Unternehmen nach den Prinzipien der Variantenkonstruktion unter Verwendung von Konstruktionskatalogen organisiert werden können.

---

[16] Branche wird in der vorliegenden Arbeit stellvertretend für eine Gruppierung von Unternehmen verwendet, die in Bezug auf ihre Informationsverarbeitung vergleichbare Anforderungen stellen. Weitere Möglichkeiten der Gruppierung, wie Unternehmensgröße, Rechtsform, Betriebstyp, Wirtschaftszweig und Produktlebenszyklus [Mertens et.al. 95] werden nicht näher erläutert, da sie aus der Sicht der semantischen Komposition dazu verwendet werden können, um „vertikale Märkte“ für den Einsatz von Anwendungssystemen zu definieren. „Branche“ wird demzufolge einem „Zielmarkt für den Einsatz von Standard-Anwendungssystemen“ gleichgesetzt.

3.3 Komponentenorientierung auf Anwendungsbereichsebene

Die auf den Begriffen des Anwendungsbereiches basierende Konstruktionsmethode, die ausschließlich Konstruktionselemente (Anwendungselemente) verwendet, die inhaltlich definiert sind, wird als ***Methode der semantischen Komposition*** bezeichnet. In den folgenden Abschnitten werden weitere Elemente der Methode (z. B. Baukastensystem, Konstruktionskatalog, semantische Relationen) und ihre Konstruktionsprinzipien ausführlich dargestellt.

3.3.1 Aufgabenorientierung

Anwendungselemente werden per Definition (siehe Abschnitt 3.1) über Aufgaben aus ihrem Anwendungsbereich identifiziert. Betrachtet man die Historie der Softwareentwicklung, so kann man hinsichtlich der Entwicklung des Schwerpunktes der Anforderungsmodellierung eine Evolution feststellen. Vor der Zeit der Datenmodellierung konzentrierte man sich auf Funktionen (Prozeduren), die in einem Programm abgearbeitet werden sollten. Danach war die Datenzentrierung treibendes Element der Softwarestrukturierung für die Umsetzung der Anforderungen. Anschließend stand die Objektzentrierung (Daten und Funktionen) im Mittelpunkt des Interesses. Allerdings zeigte sich, dass sich Anwender nicht für Lösungen bzw. Lösungselemente interessieren, sondern ihre Tätigkeiten sollten im Zentrum stehen. Die Prozessorientierung wird aktuell dafür eingesetzt und soll dieses Bedürfnis befriedigen.

Prozessorientierung muss jedoch differenziert betrachtet werden. Spezifischer sollte man von Geschäftsprozessen und Arbeitsabläufen sprechen [Lehmann/Ortner97, 62ff]. Geschäftsprozesse sind Elemente der Unternehmensplanung und –organisation. Sie können als Wertschöpfungsketten verstanden werden und definieren das eigentliche „Geschäftsmodell“ eines Unternehmens. Arbeitsabläufe dagegen sind Abfolgen von Arbeitsschritten, die unter dem Begriff Workflow erfasst werden.

Ansätze, die Spezifikation und Konfiguration (Customizing) von betriebswirtschaftlichen Standardanwendungen mittels Geschäftsprozessbausteinen umzusetzen, haben sich im Umfeld des Standardsoftware-Produktes R/3 der SAP AG, Walldorf etabliert. Unterstützt werden diese Ansätze auch durch die Wirtschaftsinformatik aus dem Saarland mit dem ARIS-Konzept [Scheer94, Keller/Popp96] und der Universität Bamberg in Ausarbeitung des Semantischen Objektmodells (SOM) [Sinz96, Ferstl/Sinz94].

Allerdings ist diese Form der Geschäftsprozessmodellierung sehr stark auf den Kontrollfluss des Arbeitsablaufes konzentriert. Eine Definition für die geeigneten Komplexität der Bausteine zu finden oder die angeforderten Baustein-Varianten zu planen und zu realisieren, war dabei nicht das Ziel.

Für den kompositionalen Aufbau einer kundenindividuellen Anwendungslösung ist die Steuerung des Ablaufes im Endprodukt wichtig. Aufbau und Ablaufsteuerung sollten jedoch getrennt voneinander bestimmt werden. Abläufe ändern sich häufiger als die Aufgaben, die in einem Unternehmen zu erledigen sind. Eine flexible Anordnung der Reihenfolgen und eine permanente Veränderung von Abarbeitungsregeln (business rules) muss durch die Anwendungssysteme der nächsten Generation unterstützt werden. Sie sind nicht Bestandteil einer Komponentenkonfiguration, sondern separat zu realisieren. Eine zentralisierte Verwaltung von Abläufen und Geschäftsregeln sichert zudem die Redundanzfreiheit im Gesamtsystem ab. Welchen Einfluss diese separaten Systeme (Workflow-Management und Business-Rules-Management) auf die Komposition der Anwendungselemente nehmen, wird in Abschnitt 3.3.2 ausführlicher dargestellt.

Abläufe werden nicht nur über festgelegte oder ad hoc entstehende Ablaufkontrollmechanismen im Anwendungssystem gesteuert, sondern sind auch Teil der Mensch-Maschine-Interaktion zwischen dem Anwender und dem Anwendungssystem. Insofern ist ein flexibler Aufbau eines User Interfaces und der dazugehörenden Menü- und Aufrufsteuerung ebenfalls ein wichtiger Faktor für die Abbildung der verschiedenen Systemanforderungen unterschiedlicher Anwenderunternehmen mit unterschiedlichen Unternehmenszielen, Wertschöpfungsketten und Märkten.

Aufgabenorientierte Bausteine für betriebswirtschaftliche Standard-Anwendungssysteme haben den Vorteil, dass sie sich an dem Sprachgebrauch des Anwendungsbereiches orientieren und dass sie die Konstruktionselemente für den Aufbau von Arbeitsabläufen und in weitergehender Aggregation auch von Geschäftsprozessen, bilden. Im Taligent-Projekt[17] von IBM und Apple wurde ein aufgabenorientiertes (task-centered) Interaktionsmodell für die Arbeit der Menschen mit einem Computer definiert. Dieses damals neue Paradigma ging davon aus, wie Menschen normalerweise

[17] Taligent wurde vollständig in die IBM integriert.

arbeiten. Sie wollen sich mehr auf ihre Aufgabenerfüllung konzentrieren als auf die Anwendungsprogramm-Manipulation [Taligent94]. Dieses Konzept wurde in der Opendoc-Technologie von IBM in Form von task-specific compound documents [Orfali/Harkey95] weitergeführt.

Der tätigkeitstheoretische Ansatz von Dahme und Raeithel [Dahme/Raeithel97] schließt die Aufgabenorientierung mit ein und hebt gerade ihre Fähigkeit zur Kommunikationsverbesserung zwischen Entwicklern und Anwendern hervor.

Dahme und Raeithel haben folgende Probleme bei der Kommunikation bezüglich der Anforderungen an ein Anwendungssystem erkannt:

- Anwender wissen bei Projektbeginn nicht genau, was sie wollen.

Eine Änderung der Anforderungen sollte jederzeit möglich sein (auch später im laufenden Betrieb des Systems). Dazu ist jedoch eine gemeinsame Kommunikationsbasis zwischen Entwicklern und Anwendern notwendig.

- Anwender sind nicht unbedingt in der Lage, das, wovon sie genau wissen, dass sie es wollen, vollständig mitteilen zu können.

Anwender sollten davon befreit sein, ihre Anforderungen an ein System beschreiben zu müssen. Ein besserer Ansatz ist die Präsentation geeigneter Lösungen in einem Katalog, wobei die Anwender (oder Anwendungsbereichsspezialisten (siehe Abschnitt 3.3.5)) durch reine Auswahl ihr Gesamtsystem komponieren können.

- Entwickler sind nicht in der Lage, ausreichend zu verstehen, was die Anwender mitteilen konnten.

Es muss ein Beschreibungssystem geschaffen werden, welches technologieunabhängig in geordneten Strukturen die Anforderungen für neue Lösungen festhält, die nicht im Katalog enthalten sind.

- Anwender verstehen in der Kommunikation mit Entwicklern nicht, was ein System noch leisten kann.

Im Angebotskatalog müssen für die Anwender Variationsmöglichkeiten präsentiert werden, die sie so nicht hätten formulieren können. Einerseits waren ihnen die

Möglichkeiten nicht bewusst, andererseits waren ihnen die Kombinationen nicht geläufig.

Anwendungselemente bilden die Plattform für eine Kommunikationsbasis zwischen den an einem Anwendungskonstruktionsprozess beteiligten Parteien.

Im Anwendungsbereich finden wir viele verschiedene Unternehmen aus unterschiedlichen Branchen, Ländern und Märkten, die hinsichtlich ihrer elementaren Aufgabenstruktur teilweise dieselben Tätigkeiten ausüben und deshalb auf dieselben Anwendungselemente zugreifen wollen. Jede dieser Anwenderorganisationen erstellt jedoch ihren eigenen Anwendungsbauplan für ihre individuelle Anwendungssystemkomposition. Neben der reinen Anwendungselemente-Auswahl gibt es noch weitere beeinflussende Faktoren, die in Abbildung 17 dargestellt sind. Als Lösung für das Problem der mitarbeiterindividuellen Gestaltung des Anwendungssystems wurde das Konzept des „benutzerspezifischen Arbeitsplatzes“ eingeführt. Dieses Konzept wird im Abschnitt 4.2 bei der Darstellung des Konstruktionsvorganges (Variantenkonstruktion) und seiner Implementierungsvorschriften noch weiter ausgeführt.

Auf der Entwicklerebene bilden die Anwendungselemente eine Architektur, die durch die Herstellung der Komponenten umgesetzt wird. Im Entwicklungsbereich muss die Zusammenführung aller Komponenten zu einem funktionsfähigen Gesamtsystem verantwortet werden. Die Zerlegung des Gesamtsystems in aufgabenorientierte Elemente ist dabei eine konkrete Anforderung. Entwickler haben es, nachdem die Anwendungselemente-Architektur einmal festgelegt ist, im Wesentlichen nur noch mit Anforderungen bezüglich der Variation von Anwendungselementen und neuen Möglichkeiten ihrer Kombination zu tun[18].

Ein bestehendes Baukastensystem für ein Standard-Anwendungssystem wird sich nur noch nach den Prinzipien der Evolutionstheorie (Abstammungslehre) weiterentwickeln:

[18] Diese Form der Weiterentwicklung bezieht sich auf denselben Anwendungsbereich. Erweiterungen des Anwendungsbereichs bedingen eine Erweiterung der Architektur respektive der Plattform.

Systementwicklungsbezeichnung		Evolutionsfaktoren der Deszendenztheorie [Duden96]
Variation	–	Mutation
Neukombination	–	Rekombination
Bewertung	–	natürliche Auslese
Orthogonalisierung	–	Isolation

Tabelle 3: Weiterentwicklung von Baukastensystemen

Wobei diese Entwicklungsschritte, als ein immer wiederkehrender Kreislauf zu betrachten sind.

3.3.2 Einflüsse auf die semantische Komposition von Anwendungssystemen

Ein betriebswirtschaftliches Anwendungssystem hat den Zweck, Unternehmen in ihren Informationsverarbeitungsprozessen zu unterstützen bzw. diese Prozesse eigenständig durchzuführen. Für die Einführung und für laufende Anpassungsmaßnahmen an veränderte Geschäftsbedingungen und –ziele ist eine sinnvolle Reihenfolge in der Varianten- und Anpassungskonstruktion des Anwendungssystems festzulegen. Die folgende Grafik zeigt, in welcher Reihenfolge dabei Vereinbarungen getroffen werden und Konstruktionshandlungen durchzuführen sind. Sie ist nicht als sequentielles Arbeitsprogramm misszuverstehen – Rücksprünge sind möglich. Die Reihenfolge orientiert sich am Grad der Beständigkeit (Konstanz) der Ergebnisse. Die Beständigkeit der Ergebnisse lässt Rückschlüsse auf die gewünschte Flexibilität des Anwendungssystems zu.

Abbildung 16: Vorgehensweise für das Einrichten eines Anwendungssystems

Durch die Darstellung der Vorgehensweise zur Einrichtung und Betreuung eines betriebswirtschaftlichen Anwendungssystems wird ersichtlich, welche Anforderungen an eine komponentenorientierte Anwendungssystementwicklung zu stellen sind.

1. **Was wollen wir tun?**
   Diese Frage muss sich ein Unternehmen immer zuerst stellen, bevor ein Anwendungssystem eingeführt oder verändert werden kann. Als Ergebnis dieser Entscheidungsfindung sollte als detaillierteste Struktur eine Vereinbarung der einzelnen Teilaufgaben, die zu bearbeiten sind, vorliegen (Aufgabenplan).

2. **Wie wollen wir es tun?**
   Diese Frage ist für die Suche nach den geeigneten Lösungen für die jeweiligen Teilaufgaben gedacht. Die Auswahl der zu verwendenden Anwendungselemente aus einem Sortiment beantwortet diese Frage. Die Vorgaben dafür entsprechen den Vereinbarungen, die als Antwort auf die Frage 1) getroffen wurden. Das Ergebnis dieser Konstruktionshandlung ist der Anwendungsbauplan für eine konkrete Kundenkonfiguration.

3. **Welche Reihenfolge ist maßgebend?**
   Mit dieser Frage werden die Arbeitsabläufe und Abarbeitungsregeln angesprochen. Die Ergebnisse dieser Fragestellung haben ebenfalls Einfluss auf die Auswahl der Anwendungselemente, sie sollten jedoch in eigenständigen Systemen verwaltet werden. Denn zum einen ist es wahrscheinlich, dass sich Abläufe und Geschäftsregeln schneller ändern als die zu erledigenden Aufgaben. Zum andern sollten die Einflüsse derartiger Vereinbarungen auf die Auswahl der möglichen Anwendungselemente durch diese Systeme konsistent gehalten werden und den Auswahlvorgang so wenig wie möglich beeinflussen.

4. **Wer soll es tun?**
   Die Frage spricht den Bereich der Arbeitsorganisation an, der den größten Wandel erfährt. Mitarbeiter werden neu eingestellt oder sie bekommen neue Aufgaben bzw. Vertreterregelungen werden überarbeitet (Urlaub, Krankheit etc.). Diese Anforderungen sollten ebenfalls in eigenständigen Systemen verwaltet werden. Ihre Einflüsse auf die Komponentenstruktur des Anwendungssystems sind systemgestützt konsistent zu halten.

Die Ausführungen haben gezeigt, dass es neben den Vereinbarungen, welche Aufgaben mit einem Anwendungssystem zu erledigen sind, noch weitere systembeeinflussende Faktoren gibt, die für ein anforderungsgerechtes Anwendungssystem festgelegt werden müssen. Diese Einflüsse werden in ihrer Wirkung auf eine semantische Komposition berücksichtigt.

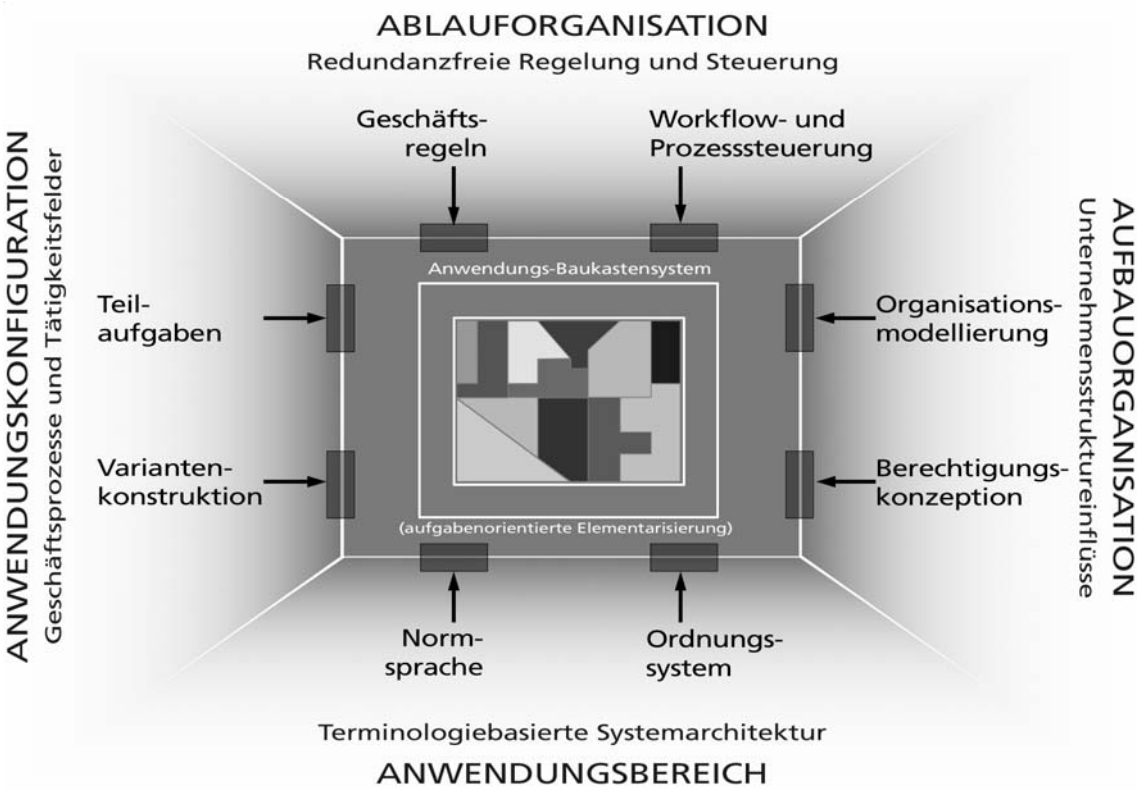


Abbildung 17: Übersicht der Einflüsse auf eine komponentenorientierte Anwendungssystementwicklung

Abbildung 17 stellt die genannten Einflüsse auf die Struktur einer Anwendungssystem-Konstruktionsumgebung dar, wie sie in den nachfolgenden Ausführungen dieser Arbeit im Einzelnen noch entwickelt und erläutert wird.

3.3.3 Beispiel-Anwendungselemente

Ein Beispiel für die Präsentation aller gestellten Anforderungen und entwickelten Fähigkeiten einer Kompositionsmethode für Anwendungselemente zu beschreiben, ist nicht einfach. Einerseits muss das Beispiel eine praktische Relevanz besitzen und darf andererseits nicht zu umfangreich sein, damit es überschaubar bleibt. Hinzu kommt, dass es immer verschiedene Möglichkeiten gibt, ein Problem zu lösen. In dem hier

vorgestellten Beispiel sind nicht alle Lösungsmöglichkeiten für die gestellten Aufgaben enthalten.

Das Beispiel stellt einen Bestellprozess unter folgenden Rahmenbedingungen dar:

- Branche: Handels- und Produktionsunternehmen
- Kundenstruktur: Direktverkauf an Endkunden (Verbraucher) und Großhandel mit Firmenkunden (Einzelhändler)

Der Geschäftsprozess „Zubehörverkauf“ besteht aus folgenden Teilaufgaben:

- Kundendaten editieren[19]
- Kundenbestellungen editieren
- Bonität prüfen
- Lagerbestände verwalten
- Produktionsplanung durchführen
- Lieferantendaten editieren
- Lieferantenbestellungen editieren
- Auftragsbestätigungen erfassen
- Lieferantenrechnungen erfassen und buchen
- Lieferscheine erfassen
- Kundenrechnungen erstellen und buchen
- Zahlungseingänge prüfen

Anhand eines Anwendungselementes soll eine mögliche Variantenbildung durch Festlegen der Variabilitätsfaktoren gezeigt werden.

[19] Editieren steht hier stellvertretend für erfassen, ändern und löschen.

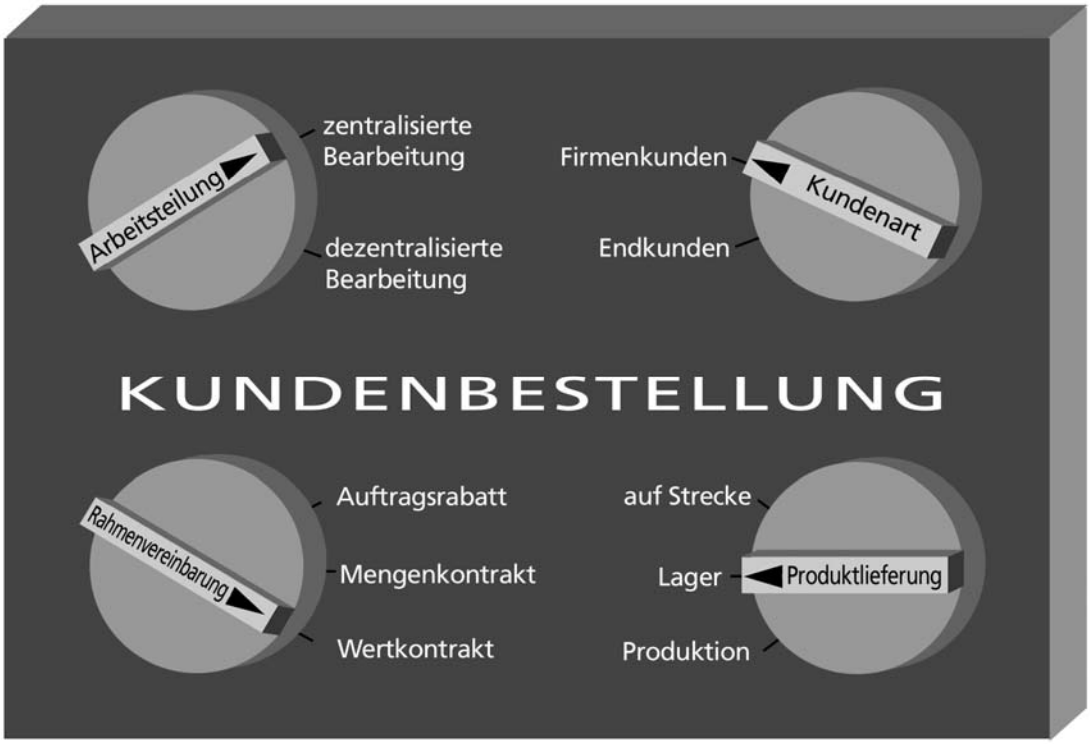


Abbildung 18: Variabilitätsfaktoren des Anwendungselementes *Kundenbestellung*

Es werden nun zwei exemplarische Arbeitsabläufe des Geschäftsprozesses „Zubehörverkauf“ vorgestellt. Für die ***Methode der semantischen Komposition*** ist nur die Zusammenstellung der Anwendungselemente von Interesse (Zusammenbauplan). Die durch ein Workflow- und Prozess-Management-System umgesetzte Ablaufsteuerung wird separat zur Komposition der geeigneten Anwendungselemente festgelegt und ist dann in gewissen Grenzen auch später noch flexibel veränderbar (in Abstimmung mit der Festlegung von Geschäftsregeln und in Übereinstimmung mit dem Berechtigungskonzept). Mögliche Darstellungsmittel (z.B. „Ereignisgesteuerte Prozessketten“[20]) sowie Anwendungshinweise und Beispiele können der entsprechenden Literatur entnommen werden [Jablonski et.al 97, Scheer94, Scholz/Volmering04].

[20] Es soll hier keine Diskussion geführt werden, ob Arbeitsvorgangsmodelle zur Modellierung von Workflows geeignet sind (vgl. hierzu [Jablonski et. al. 97, 80ff.]).

Für die Komposition eines „Streckengeschäftes für Privatkunden“ in einem Handelsunternehmen für Motorradzubehör werden folgende Anwendungselemente ausgewählt (Teilaufgabe in [ ]):

- Privatkundenstamm [Kundendaten editieren]
- Kundenbestellung auf Strecke (Eigenschaften: Produktlieferung auf Strecke, Kundenart Endkunden, Arbeitsteilung zentralisierte Bearbeitung, Rahmenvereinbarung keine) [Kundenbestellungen editieren]
- Lieferantenstamm [Lieferantendaten editieren]
- Lieferantenbestellung [Lieferantenbestellungen editieren]
- Lieferantenrechnung [Lieferantenrechnungen erfassen und buchen]
- Privatkundenrechnung [Kundenrechnungen erstellen und buchen]
- Offene Posten-Verwaltung [Zahlungseingänge prüfen]

In demselben Unternehmen wird noch eine „Lagerbestellung für Firmenkunden“ komponiert:

- Firmenkundenstamm [Kundendaten editieren]
- Kundenbestellung für Lagerartikel (Eigenschaften: Produktlieferung Lager, Kundenart Firmenkunden, Arbeitsteilung zentralisierte Bearbeitung, Rahmenvereinbarung Auftragsrabatt) [Kundenbestellungen editieren]
- Bonitätsprüfung [Bonität prüfen]
- Lagerbestandsverwaltung [Lagerbestände verwalten]
- Lieferschein [Lieferscheine erfassen]
- Firmenkundenrechnung [Kundenrechnungen erstellen und buchen]
- Offene Posten-Verwaltung [Zahlungseingänge prüfen]

Zu den jeweiligen Teilaufgaben wurden die entsprechenden Anwendungselemente ausgewählt. Die Unterstützung der Anwendungselemente-Auswahl ist der Sinn und Zweck der ***Methode der semantischen Komposition***. Die Bezeichner der Anwendungselemente sind eindeutig. Anwendungselemente werden damit in einem Bauteil-

Repository verwaltet. Jedes dieser Anwendungselemente (nicht nur die Kundenbestellung) hat Eigenschaften, die für die Auswahl verwendet werden.

Als Konfliktmöglichkeit für das dargestellte Beispiel seien hier die Eigenschaften der Kundenbestellung in Bezug auf das Merkmal „Produktlieferung“ genannt. Es gibt grundsätzlich zwei Möglichkeiten diese Systemvariabilität abzubilden:

1. Für die drei genannten Variabilitäten wird jeweils ein Anwendungselement zur Auswahl bereitgestellt.
2. Es existiert ein so genannter Multigegenstand (siehe Abschnitt 3.4.2), der verschiedene Eigenschaften eines Merkmals in sich vereinigt. Es ist also ein Anwendungselement im Sortiment enthalten, welches alle drei Produktlieferungsfähigkeiten besitzt.

Das Anwendungselement der zweiten Kategorie kann alle Bestellaufträge verarbeiten, egal ob sie nun „auf Strecke“ oder vom Lager geliefert werden oder ob sie erst noch produziert werden müssen. Die Frage, ob die Produktlieferung vom System selbst erkannt wird (z.B. anhand der Artikelnummer) oder ob der Anwender über eine Auswahl bei der Erfassung (radio-button, checkbox o.ä.) die Produktlieferung bestimmt, ist dabei unerheblich.

In jedem Fall ist die Entscheidung für die erste oder zweite Kategorie eine Modellierungsentscheidung in Bezug auf die Zerlegung des Gesamtsystems in (relativ) unabhängige Bauteile. Beide Varianten haben Vor- und Nachteile. Für Multigegenstände ist es wahrscheinlicher, dass eine Erweiterung der Eigenschaften mit einem höheren Aufwand verbunden ist als bei einer Variantenbildung von Anwendungselementen der Kategorie 1. Die Aufgabenstellung kann auch Multigegenstände erfordern. Dann stehen die Multigegenstand-Anwendungselemente neben den Anwendungselementen mit nur einer zugeordneten Eigenschaft gleichwertig im Sortiment.

Der Konfliktfall soll zeigen, dass über jede Modellierungsentscheidung des Beispiels diskutiert werden kann. Es ist jedoch dafür „nur“ ein Beispiel und soll in den folgenden Ausführungen die Ideen und Lösungsansätze der ***Methode der semantischen Komposition*** erläutern.

Ein tiefer gehendes Verständnis für die innere Struktur eines Anwendungselementes wird das durchgängige Beispiel in Abschnitt 4.1.3 (für die „Komposition einer Beispielanwendung“ in ***TKMS***) über alle Abstraktionsebenen einer *Lagerbestellung* erzeugen.

3.3.4 Konstruktionsprinzipien

Man sieht, wie man bei dem Versuch, sich in die Höhen der Abstraktion zu erheben, vielmehr in den Sumpf von Lehrmeinungen gerät.

PAUL LORENZEN[21]

Die hier vorgestellte Methode ist ein konstruktiver Ansatz. Dies bedeutet, dass alle elementaren Konstruktionselemente ausdrücklich definiert und eingeführt werden müssen [Wedekind81, 114, Lorenzen74]. Hierbei ist der Begriff „Konstruktionselement“ auch auf die normierten Fachbegriffe (Termini) eines Anwendungsbereiches auszudehnen. Diese Fachbegriffe spezifizieren die Eigenschaften von Anwendungselementen und sind dadurch Teil des Konstruktionsvorganges. Nur aus inhaltlich präzise definierten Bausteinen (Termini und Anwendungselemente) werden komplexere Gebilde (Baugruppen) erstellt (konstruiert). Weil diese auf den Begriffen des Anwendungsbereiches basierende Konstruktionsmethode ausschließlich Konstruktionselemente (Anwendungselemente) verwendet, die inhaltlich definiert sind, wird sie als ***Methode der semantischen Komposition*** bezeichnet.

Die Vorteile einer konstruktiven Methode liegen darin, dass alle Konstruktionshandlungen begründet, nachvollziehbar, verifizierbar und verständlich sind. Ziel ist es nicht, ein reines Abbild (Modell) der Wirklichkeit zu schaffen, sondern die wesentlichen Elemente eines realen Weltausschnittes zu erkennen und daraus die Modelle und Systeme zu konstruieren.

Für eine komponentenorientierte Anwendungssystementwicklung lassen sich verschiedene Konstruktionsprinzipien anwenden. Eine Übersicht der geeigneten Prinzipien mit ihren Erläuterungen wird im Folgenden dargestellt.

---

[21] Zitiert nach [Wedekind92, 15]

3.3.4.1 Abstraktion - Konkretion

In Ergänzung zu den Ausführungen aus Abschnitt 2.1.2.2 ist die Abstraktion, basierend auf der Gleichheit als Fundierungsrelation, in verschiedene Grundtypen zu unterscheiden [Ortner83, 76ff, Wedekind92, 19ff, Ortner97, 121ff, Schienmann97, 56ff]:

- ***Subsumtion*** ist ein begriffsbildender Abstraktionsvorgang. Dabei werden intensional durch das Finden von invarianten (gemeinsamen) Eigenschaften (Merkmalen) von Gegenständen neue, übergeordnete Begriffe gebildet (Prädikation). Extensional lassen sich Gegenstände zu Klassen zusammenfassen. Die Klasse entspricht dem Begriff, die Gegenstände seinen Instanzen.

- ***Subordination*** ist ein hierarchiebildender Abstraktionsvorgang. Durch eine schrittweise Trennung von invarianten Merkmalen und variierenden Merkmalen werden übergeordnete Klassen der Begriffe festgelegt (Begriffsinklusion). Die so festgelegten Hierarchieebenen entsprechen der Generalisierungs/Spezialisierungs- oder Art/Gattungsbeziehung.

- ***Identität*** stellt in Bezug auf die Anwendungssystementwicklung eine besondere Form der Abstraktion dar. Die Identität repräsentiert durch viele (schrittweise) Wechsel der Sprachebenen von der fachlichen Ebene hin zur Implementierung die Lösungskonkretion. Nun ist eine Software in den wenigsten Fällen von der fachlichen Spezifikation bis hin zur Implementierung nur durch schrittweise Konkretion ohne ein Zerlegen (Partition) in Bestandteile realisierbar. Das Beinhalten von kompositionalen Aspekten in der Lösungskonkretion bedingt auch die Bezeichnung der Komplexionsebenen (Kompositionsebenen) als Abstraktionsebenen. Nicht zu verwechseln ist dies mit dem von Wedekind [Wedekind92, 48] angedeuteten Missverständnis einer veralteten Form der Abstraktion, der isolierenden Abstraktion, bei der durch Weglassen von unwesentlichen Merkmalen eine Abhängigkeit als Äquivalenz fehlgedeutet wird.

Alle Abstraktionshandlungen führen zu einem Wechsel der Sprachebene. Die Subordination entspricht von ihrer Handlung her der Subsumtion. Es werden die durch Prädikation gebildeten Begriffe nach derselben Methode zu „neuen“ Begriffen

generalisiert. Wichtig ist dabei, dass alle Objekte (Gegenstände oder Begriffe) einer untergeordneten Hierarchiestufe äquivalent zueinander sind – es besteht eine Äquivalenzrelation (Austauschbarkeit) zwischen allen unter einen Begriff „fallenden" Objekten.

3.3.4.2 Komposition - Partition

Auch für die Komposition, basierend auf der Abhängigkeit als Grundbeziehung, ist gegenüber Abschnitt 2.1.2.2 eine differenzierte Betrachtung sinnvoll. In Schienmann wurde eine sehr umfangreiche Übersicht aller Kompositionsarten (Teil/Ganze-Beziehungsarten) erstellt. Dabei wurden sechs Beziehungsarten unterschieden [vgl. Schienmann94, 52f]:

- Verknüpfung (z.B. Ehemann, Ehefrau < Ehe)
- Verschmelzung (z.B. Eier, Mehl < Kuchen)
- Ganzheit (z.B. Motor, Getriebe < Antrieb)
- Zusammenstellung (z.B. Spieler < Mannschaft)
- Behältnis (z.B. Apfel < Korb)
- Zuordnung (z.B. Auftrag < Kunde)

Schienmann hat dabei die Kriterien **Anteil** der Teile am Ganzen, **Bindung** (Kopplung) zwischen den Teilen eines Ganzen und **Verwandtschaft** der Teile mit dem Ganzen untersucht. Für eine komponentenorientierte Anwendungssystementwicklung auf Herstellungsebene lassen sich sicher für alle Kompositionsarten Anwendungsfälle finden. Für die Komposition von Anwendungselementen sind nur die Ganzheit und die Zusammenstellung von Bedeutung.

Für die **Ganzheit** (Integrativ) gilt dabei, dass die Teile einen essentiellen oder zumindest wichtigen Anteil am Ganzen darstellen – wichtig in Bezug auf eine bestimmte Eigenschaft (bzw. in Bezug auf die Existenz). Die Ganzheit besitzt eine hohe Bindung. Bezüglich der Verwandtschaft - Schienmann hat dies als Instanziierung der beteiligten Objekte von demselben Objekttyp bezeichnet - ist bei der Ganzheit alles möglich: verwandte Teile, verwandtes Ganzes und keine Verwandtschaft. Anwendungselemente besitzen keine Vererbungsbeziehungen, insofern ist eine Verwandt-

schaft nur zwischen Anwendungselementen zu definieren, die aufgrund ihrer Zusammenfassung zu einer Sachmerkmalleiste (siehe Abschnitt 3.4.2) über dieselben Merkmale verfügen. Eine Verwandtschaft zwischen Teilen und Ganzheiten ist dadurch nicht möglich.

Bei einer **Zusammenstellung** von Anwendungselementen ist der Anteil am Ganzen unwesentlich. Es existiert zwischen den Teilen eine geringe bis gar keine Bindung. Eine Verwandtschaft der Teile ist möglich.

Für beide Kompositionsarten gilt, dass immer ein (funktionales) Zusammenwirken der Teile in Bezug auf das Ganze existieren muss [Martin/Odell95, 197], wobei die Integration der Einzelteile im Zusammenwirken für die Ganzheit sehr ausgeprägt ist und bei der Zusammenstellung eine Unabhängigkeit (Orthogonalität) zwischen den Teilen vorherrscht.

Die Handlung der Komposition, d.h. die Baugruppenbildung muss sich diesen Beziehungen zwischen Teilen und dem Ganzen unterwerfen. Das Ordnungssystem der ***Methode der semantischen Komposition*** sieht hierfür eine Separierung der inhaltlichen Beziehungen zwischen Anwendungselementen vor (semantische Relationen). Regeln für den Aufbau und die Anwendung der semantischen Relationen werden im Abschnitt 3.4.4 ausführlich dargestellt.

Die Partition bzw. Elementarisierung eines Systems wird in dieser Arbeit mit der Zielsetzung der Entwicklung eines Baukastensystems gesehen. Für eine geeignete Zerlegung müssen mehrere Aspekte berücksichtigt werden, welche die verschiedenen Kompositionshandlungen positiv beeinflussen:

Integrationsaspekt

Aus der Perspektive der Anwendungselemente-Komposition muss es ein Bestreben geben, die Teile einer Ganzheit aus ihrer integrativen Vernetzung herauszulösen und dabei die Bindungen zu reduzieren. Dadurch werden die Teile unabhängiger voneinander. Erst dann wird es möglich, sie frei zu einer Zusammenstellung zu kombinieren. Für eine Reduzierung der Abhängigkeiten zwischen Anwendungselementen sind zuerst ihre Integrationsaspekte in ein Gesamtsystem zu identifizieren und zu analysieren:

- **Ablaufintegration:** Durch eine Separierung der Ablaufsteuerung und der Geschäftsregeln von den Anwendungselementen werden die Abhängigkeiten zwischen den in Abläufen angeordneten Anwendungselementen reduziert. Die Wiederherstellung der Integration erfolgt erst nach dem Zusammenstellen der an einem Arbeitsablauf beteiligten Anwendungselemente durch ein Regel- und Steuerungssystem (Workflow-/Prozess-Management-System und Geschäftsregel-Verarbeitungs-System[22]). Dadurch ist eine höhere Flexibilität bei der Definition und Änderung von Arbeitsabläufen sichergestellt.

- **Funktionsintegration:** Die Auswirkungen der Funktionsintegration lassen sich nur durch das Erkennen der wesentlichen Elemente (Faktoren) von Funktions- und Ablaufstrukturen reduzieren[23]. Die Faktoren stellen (möglichst) unabhängige Elementarbausteine dar, die durch sinnvolle Kombination komplexe Funktionen und damit komplexe Abläufe erstellen. Sind auf der inhaltlichen Seite der Anwendungselemente integrative Beziehungen vorhanden, die sich nicht auflösen lassen, so müssen sie mit möglichst wenig Aufwand im System abgebildet werden. Die erstellten Lösungen müssen diesbezüglich konsistent gehalten werden.

- **Datenintegration:** Durch den Einsatz einer Anwendungsplattform als Integrationsbasis wird die Datenintegrität weitgehend sichergestellt. Anwendungselemente sind dann in der Regel nicht direkt für eine physische Datenspeicherung zuständig. Die Business-Objekte und Services der Plattform sind verantwortlich für eine korrekte und konsistente Datenablage (Persistenz).

---

[22] An dieser Stelle soll keine Aussage getroffen werden, ob das Workflow-Management-System die Geschäftregeln ebenfalls verarbeitet oder dafür ein separates System in Zusammenarbeit mit dem Workflow-Management-System eingesetzt wird.

[23] Es geht hierbei nicht um die Redundanzfreiheit von verwendeten Funktionen (Methoden) im Gesamtsystem, die durch das Anbieten der Methoden in einem Entwicklungs- und/oder Laufzeit-Framework ermöglicht werden kann, sondern um die inhaltlichen, funktionalen Abhängigkeiten zwischen Anwendungselementen.

Kohärenz als Modellierungskriterium

Die Eigenschaften, die eine Komponente anbietet, sollten in einem begründeten Zusammenhang stehen. Der Grad der Abhängigkeit wird durch die Kohärenz ausgedrückt [Ortner97, 130f]. Abhängige Eigenschaften sollten möglichst in einem Anwendungselement vereinigt sein. Das Ziel ist also eine hohe Kohärenz. Im Endeffekt stellen die Anwendungsbereiche, generalisiert für ihre Branchen, die Anforderungen an die Zusammenstellung der Eigenschaften für Anwendungselemente und Baugruppen dar (Referenzmodelle). Eine falsche Konzeption von Anwendungselementen liegt dann vor, wenn für die Erledigung einer (elementaren) Aufgabe mehrere Anwendungselemente benötigt werden, die zusammen weit mehr Eigenschaften anbieten, als unbedingt benötigt werden. Bei der ***Methode der semantischen Komposition*** kann die Kohärenz der Baugruppen jederzeit korrigiert werden. Die richtige Kohärenz von elementaren Anwendungselementen muss jedoch sorgfältig entwickelt werden. Anhaltspunkte dafür liefert das Verfahren der nachfolgend dargestellten Orthogonalisierung.

Kopplung

Kopplung oder Bindung ist ein Gradmesser für die Unabhängigkeit von Komponenten. Es ist ein erklärtes Ziel, eine möglichst geringe Bindung zwischen Anwendungselementen zu erzeugen. Für die Bindung in Bezug auf Baugruppen gibt es zwei Kopplungsrichtungen: Bindung innerhalb einer Baugruppe und Bindung zwischen Elementen verschiedener Baugruppen. Eine hohe Integration innerhalb einer Baugruppe vermindert die Bindung zu anderen Anwendungselementen oder Baugruppen [Schienmann94, 51]. Damit ist die Kohärenzforderung erfüllt, allerdings lassen sich die Komponenten im Sinne der Baukastenkonstruktion nicht mehr mit der nötigen Flexibilität, d.h. mit den gewünschten Freiheitsgraden kompositiv zu neuen, angepassten Lösungen kombinieren.

Hierbei werden die Konflikte deutlich, die bei der Zerlegung von Gesamtsystemen in ihre Elemente, entsprechend dem Baukastenprinzip, entstehen. Zum einen soll die Kohärenz hoch sein und zum anderen soll die Bindung sowohl innerhalb als auch zwischen Baugruppen (außerhalb) möglichst gering sein. Folglich sollte auch die Integration, zumindest aus der Kompositionssicht, möglichst gering sein.

Auf der Kommunikationsebene hat Börstler die Ziele hoher Kohärenz und geringer Bindung folgendermaßen definiert[24] [Börstler94, 14]:

- Jede Komponente kommuniziert mit so wenig anderen Komponenten wie möglich.
- Zwei miteinander kommunizierende Komponenten tauschen so wenig Daten wie möglich aus.
- Jegliche Kommunikation zwischen Komponenten ist explizit erkennbar.

Damit diese Ziele erreicht werden können, ist eine größtmögliche Unabhängigkeit (Orthogonalität) und Austauschbarkeit in einem Baukastensystem für Anwendungselemente anzustreben [Schmid et.al. 95, 48].

Orthogonalisierung

Die folgenden Ausführungen beziehen sich auf endlich dimensionale Räume und es wird nur das Standard-Skalarprodukt (euklidsche Skalarprodukt) betrachtet. Als weitere Voraussetzung wird festgelegt, dass die Orthogonalität zwischen mehreren Vektoren immer paarweise zu prüfen ist.

Ihren Ursprung hat die Orthogonalisierung in der Mathematik. Dort dienen Orthogonalisierungsverfahren (Householder und Schmidt) der Lösung von linearen Gleichungssystemen; im Speziellen von Quadratmittelproblemen. Die dazu verwendeten Transformationen verhalten sich durch die besonderen Eigenschaften orthogonaler Vektoren und Matrizen in Bezug auf die Auswirkung von Rundungsfehlern numerisch günstig [Bronstein et.al. 95, Stoer89, 180ff].

Die Orthogonalität zweier Vektoren ($\vec{a} \perp \vec{b}$) *liegt dann vor, wenn gilt:* $\vec{a}\vec{b} = \mathbf{0}$*, und weder* $\vec{a}$ *noch* $\vec{b}$ *gleich dem Nullvektor sind* [Bronstein et.al. 95, 153].

[24] Börstler hat zwar von Modulen gesprochen, die Aussagen gelten jedoch ebenso für Komponenten. In Abschnitt 3.3.8 wird auf die schwache Differenzierung zwischen Modulen beim „Programmieren im Großen“ und Komponenten eingegangen.

Das bedeutet, dass orthogonale Vektoren linear unabhängig sind. Eine lineare Abhängigkeit zwischen drei Vektoren ($V_1 – V_3$) liegt dann vor, wenn $\lambda V_1 = \mu V_2 + \delta V_3$ erfüllt ist.

Beispiel für lineare Abhängigkeit:

Die Vektoren $\begin{pmatrix}3\\1\end{pmatrix}$, $\begin{pmatrix}1\\0\end{pmatrix}$ und $\begin{pmatrix}0\\1\end{pmatrix}$ sind nicht linear unabhängig, denn

$$\begin{pmatrix}3\\1\end{pmatrix} = 3 * \begin{pmatrix}1\\0\end{pmatrix} + 1 * \begin{pmatrix}0\\1\end{pmatrix}$$

Dieses Beispiel zeigt deutlich, dass es in einem n-dimensionalen Raum maximal n linear unabhängige paarweise orthogonale Vektoren geben kann.

Ein Skalarprodukt lässt sich nur von Vektoren gleicher Dimension bilden. Folglich kann man die Orthogonalität nur zwischen gleichdimensionierten Vektoren feststellen.

Mit der Orthogonalisierung wird hier das Ziel verfolgt, unabhängige Einheiten für die Aufbaustrukturen im Ordnungssystem für Anwendungselemente zu bestimmen. Die Orthogonalisierung ist hinsichtlich ihres Einsatzes für den strukturellen Aufbau des Ordnungssystems in einer besonderen Art und Weise anzuwenden. Für die Erläuterung der Orthogonalisierung müssen folgende Ergebnisse der Entwicklung eines Ordnungssystems im Sinne der ***Methode der semantischen Komposition*** (siehe Abschnitt 3.4) vorweggenommen werden:

- Sachmerkmalleisten repräsentieren einen Aufgabenbereich
- Sachmerkmalleisten werden durch einen Vektor von Merkmalen beschrieben
- Anwendungselemente werden durch Merkmalausprägungen (Eigenschaften) beschrieben. Insofern sind die zugehörigen Merkmale in Sachmerkmalleisten gruppiert die Meta-Ebene der Anwendungselemente-Beschreibung

Die Merkmalsvektoren aller Sachmerkmalleisten eines Anwendungsbereiches werden bezüglich ihrer Orthogonalität paarweise geprüft. Dabei entspricht die Anzahl der Merkmale den „Dimensionen“ des Vektors. Es ist am einfachsten die zu betrachtenden Vektoren zu digitalisieren, d.h. es werden nicht die Merkmalswerte einer Sachmerk-

malleiste mit anderen verglichen sondern ihre Zuordnung (ausgedrückt in 0 oder 1) zu einem Merkmal der Menge aller Merkmale. Damit sind auch alle zu vergleichenden Vektoren gleichdimensioniert.

Die Orthogonalisierung der Meta-Ebene der Beschreibung von Anwendungselementen dient dazu den Anwendungsbereich des Systems in möglichst unabhängige Aufgabenbereiche zu unterteilen (Partition).

Merkmale	**Sachmerkmalleiste “ Kundenangebot“**	**Sachmerkmalleiste “ Kundenbestellung“**
Bestandsführung	0	1
Produktbereitstellung	0	1
Rahmenvereinbarung	1	1
Terminierung	1	1
Arbeitsteilung	0	1
Kundenart	1	1
Opportunity-Aspekte	1	0
Provisionsverfahren	1	1
Planungsrelevanz	1	0

Tabelle 4: Beispiel für zwei digitalisierte Sachmerkmalleisten

In Bezug auf die Partition wird die Unabhängigkeit von Merkmalsvektoren der Sachmerkmalleisten durch den Besitz gemeinsamer Merkmale eingeschränkt. Diese Einschränkung wirkt sich auf die Kombinationsfähigkeit der in den Sachmerkmalleisten gruppierten Anwendungselemente aus. Mit dem „Grad der Orthogonalität“ kann die eingeschränkte Kombinierbarkeit gemessen werden.

Grad der Orthogonalität:

$$\begin{pmatrix}1\\0\\0\\0\\1\\1\end{pmatrix} \cdot \begin{pmatrix}0\\1\\0\\0\\1\\1\end{pmatrix} \cdot \begin{pmatrix}0\\0\\1\\0\\1\\1\end{pmatrix} \cdot \begin{pmatrix}0\\0\\0\\1\\1\\1\end{pmatrix}$$

Die vier Vektoren, die in das Orthogonalisierungsverfahren eingebracht werden, sind nicht vollständig orthogonal. Das bedeutet, sie sind nicht über alle Dimensionen (bzw.

Merkmale) orthogonal. Es kann jedoch ein Orthogonalitätsgrad für die unabhängigen Dimensionen bestimmt werden. In diesem Fall, bei 4 unabhängigen von insgesamt 6 Dimensionen ist ca. 66,67 % Orthogonalität vorhanden. Eine vollständige Orthogonalität ist aus wirtschaftlichen Gründen nicht immer anzustreben. Wird auf Unabhängigkeit bewusst verzichtet, so stellt dies kein Problem dar. Wichtig ist nur, dass in der Systementwicklung die Abhängigkeiten erkannt sind.

Es gibt nun die Möglichkeit für den sechsdimensionalen Raum in dem sich die oben aufgeführten Vektoren befinden, einen **Teilraum** zu bilden. Dies entspricht einem „Abspalten“ von Komponenten, die die Abhängigkeit begründen (grau hinterlegter Bereich). Dadurch werden vollständig orthogonale Vektoren hergestellt. Dieses Abspalten wird im Aufbau eines Ordnungssystems als Indiz für die Bildung eigenständiger Sachmerkmalleisten verwendet. Beispielsweise ist es denkbar für die Rahmenvereinbarung einen eigenen Aufgabenbereich zu definieren. Dies bedeutet, dass dafür eine eigene Sachmerkmalleiste mit eigenen Anwendungselementen erstellt wird.

Die Orthogonalisierung ist auf jeder Abstraktionsebene als Modellierungsrichtlinie für die geeignete Abgrenzung von elementaren Sachmerkmalleisten und Sachmerkmalleisten für Baugruppen anzuwenden. Allerdings muss sie immer innerhalb derselben Abstraktionsebene durchgeführt werden, d.h. nur Sachmerkmalleisten, die derselben Abstraktionsebene entstammen, dürfen in das Orthogonalisierungsverfahren einbezogen werden.

Methodisch betrachtet, ist die Orthogonalisierung in einem Ordnungssystem für Anwendungselemente nur approximativ. Denn eine vollständig orthogonale Aufbaustruktur kann es nicht geben. Es werden allerdings Gemeinsamkeiten der Anwendungslösungen und ihrer variablen Erweiterungen erkannt. Als Basis für das Erkennen von unabhängigen Einheiten können die Fachbegriffe einer Sprache und deren beschreibende Merkmale verwendet werden. Eine Fachsprache setzt die Erfahrung der Menschen in einem Fachbereich mit Sachverhalten der realen Welt in die Bildung von Begriffen um. Differenzierte Begriffe in Kompositionsstrukturen bezeugen einen Sachverhalt, für den eine separate (variable) Einheit benötigt wurde. Diese Einheit hat eigenständige, von anderen Einheiten abgrenzbare, Eigenschaften und ist gegebenenfalls variierbar.

Anwendungselemente unterschiedlicher Sachmerkmalleisten sollten möglichst keine voneinander abhängigen Variabilitätsfaktoren besitzen. Damit soll ausgedrückt werden, dass sich die Eigenschaften der Anwendungselemente unterschiedlicher Aufgabenbereiche nicht gegenseitig bedingen dürfen [Weide et.al. 91, 26]. Es ist deshalb wichtig, die elementaren „Faktoren“ für die Konstruktion von Anwendungselementen zu finden.

Faktorenanalyse

Auf der „Fourth International Conference of Software Reuse“ erklärte Biggerstaff [Biggerstaff96], dass die Softwarekomponentenentwicklung in ein Skalierbarkeitsproblem sowohl auf horizontaler als auch vertikaler Ebene gerät und sich deshalb ein Paradigmenwandel hin zu unabhängigen „Faktoren“ vollzieht, woraus flexibel Anwendungen durch Komposition konstruiert werden können. „...where factors are pure abstractions and pure features that can be composed in combinatorically many ways“ [Biggerstaff96, 237]. Diese Aussage wird in der ***Methode der semantischen Komposition*** absolut unterstützt.

Die Suche nach den Grundelementen (Faktoren) eines Wissensbereiches wird als Faktorenanalyse bezeichnet und wurde beispielsweise in der Bedeutungsforschung für Begriffe [Osgood73, Osgood et.al. 78], der Intelligenzdiagnostik [Dörner87, 105ff] oder sehr erfolgreich in der Marketing-Forschung (Marktforschung) [Kotler89, 645] eingesetzt. Ziel dabei ist die Ermittlung von Faktoren, die einer großen Menge verschiedener Eigenschaften zugrunde liegen. Methodisch werden dabei empirische Werte statistisch auf ihre Interkorrelation hin untersucht und die fundamentalen Zusammenhänge bzw. deren unabhängige Faktoren ermittelt.

In der Anwendungssystementwicklung wird Faktorenanalyse dadurch betrieben, dass viele verschiedene Arbeitsabläufe, Aufgaben und deren Zusammenhänge, Informationsbedürfnisse und –erzeugung sowie zukünftige Anforderungen an die Funktionalität von Systemen untersucht werden mit dem Ziel der Ermittlung von elementaren Prozessen und elementaren Eigenschaften der Anwendungselemente. Die Bildung von Systemlösungen, die diese Elementarfaktoren repräsentieren, mit denen dann konstruktiv durch Variation und Komposition Anwendungselemente und Baugruppen für Anwendungssysteme erstellt werden, kann als Fortführung der Orthogonalisierung auf der Ebene der elementaren Anwendungselemente gesehen werden. Werden auf

dieser Abstraktionsebene die Eigenschaften orthogonalisiert, so erhält man dabei die elementaren Faktoren für den Anwendungsbereich. Die so gefundenen Faktoren stellen also die Variabilitätsfaktoren der elementaren Anwendungselemente dar (siehe Beispiel Abbildung 18).

Die hier vorgestellten Konstruktionsprinzipien sollen in ihrer Anwendung den Entwurf eines terminologiebasierten Ordnungssystems unterstützen. In den nächsten zwei Abschnitten wird noch ein weiteres Gestaltungsprinzip („Wiederverwendung durch Auswahl“) für Anwendungselemente bzw. für deren Einsatz in der ***Methode der semantischen Komposition*** erläutert. Zuvor ist jedoch die Arbeitsteilung bei einer komponentenorientierten Anwendungssystementwicklung zu untersuchen, deren Ergebnis eine Begründung für das Gestaltungsprinzip liefert.

3.3.5 Arbeitsteilung bei einer komponentenorientierten Anwendungs-entwicklung

I would give the spec to marketing and say,

„Please give me your feedback. Is this the right set of features to do?“

And marketing would either read it or not read it,

because it was way too long.

Or, if they did read it, they would get lost in it,

because it`s a super-technical thing.

And if they did comment on it, ... they would say,

„Well, we think this dialog box is laid out wrong.

You should really have the check boxes on the left“, or something.

It`s not the feedback you want as a program manager.

MIKE CONTE[25]

Eine geschichtete Aufteilung einer komponentenorientierten Anwendungssystem-entwicklung in Anwendungsbereichsebene und Herstellungsebene ist nicht nur techno-

[25] Aus Microsoft secrets, zitiert nach [Zeller97, 235]

logisch begründet, sondern ist auch dadurch bedingt, dass die vielfältigen Fähigkeiten und Erfahrungen, die für einen ganzheitlichen Entwicklungsansatz erforderlich sind, nicht in Einzelpersonen vereinigt werden können. Eine ausschließlich vertikale Arbeitsteilung ist nicht mehr ausreichend. Mit fortschreitender Lösungskonkretion werden jeweils andere Eigenschaften der Problemlösungen fokussiert [Dreibholz75, 234f]. Es müssen folglich auf unterschiedlichen Abstraktionsebenen unterschiedliche Entwickler entsprechend ihrer Fähigkeiten beteiligt werden. Ein geschichteter Softwareentwicklungsprozess zeichnet sich also durch Kunden-Lieferanten-Verhältnisse auf vielen Abstraktionsebenen aus. Innerhalb einer Ebene können jeweils Lieferanten und Lieferprodukte (Angebot) sowie Kunden und Kundenanforderungen (Nachfrage) identifiziert werden.

In Abschnitt 3.1 wurden zwei Abstraktionsbereiche definiert, die sehr unterschiedliche „Endprodukte“ erstellen wollen:

1. **Anwendungsbereichsebene**, dort werden Anwendungssysteme komponiert.
   Dieser Konstruktionsbereich, der Anwendungselemente „verwendet“, zeichnet sich dadurch aus, dass alle Komponenten dieser Ebene durch die Aufgabensicht ihres Anwendungsbereiches bestimmt sind.

2. **Herstellungsebene**, das Ziel hierin ist die Entwicklung von Anwendungselementen unter Berücksichtigung ihres Zusammenspiels.
   Die zweite Ebene ist ein Bereich, in dem Herstellungselemente (Creatorelemente) für Anwendungselemente existieren und produziert werden. Das oberste Ziel für die Existenz jedes Creatorelementes ist es also, auf irgendeiner Kompositionsstufe in ein Anwendungselement einzugehen.

In jeder der beiden Ebenen gibt es Arbeiter, die mit den vorhandenen „Rohstoffen“ neue Produkte herstellen. In der Herstellungsebene werden mit vorhandenen und neu produzierten Creatorelementen neue (elementare) Anwendungselemente hergestellt oder Varianten von bereits bestehenden Anwendungselementen gebildet. In der Welt der Anwendungselemente werden aus existierenden Anwendungselementen Anwendungssysteme komponiert. Die Arbeiter werden entsprechend ihren Tätigkeiten **Creator** (Herstellung von Anwendungselementen aus Creatorelementen) und

Composer (Komposition von Anwendungssystemen aus Anwendungselementen) genannt[26] [Ortner et.al. 99, 254, Kalkmann et.al. 96, 16]

Diese Bezeichnungen stehen in der Softwareentwicklung natürlich nur für Tätigkeits-Rollen. Ein Entwickler kann sowohl als Creator als auch als Composer tätig sein. Als Composer treten nicht nur Entwickler auf, sondern auch Berater, Anwendungs-experten, Organisationsspezialisten und Wissensingenieure der Anwender-unternehmen.

Im Bereich der Creatorelemente gibt es wiederum viele verschiedene Ebenen, die ebenfalls durch Kunden-Lieferanten-Verhältnisse ausgezeichnet sind. Alle Software-artefakte die zur Wiederverwendung geeignet sind, können anderen Entwicklern, die entweder komplexere Creatorelemente oder Anwendungselemente herstellen, für eine Komponentenfertigung zur Verfügung gestellt werden (Ebene der Komponenten-produktion). Sind Komponenten ohne Anpassung wieder zu verwenden, so ist auf der Creatorebene im Normalfall eine Programmierung von Verbindungssoftware (Script oder Kitt) notwendig, damit die Komponenten interagieren können (Montageebene).

Stellen Creators Anwendungselemente her, das Endprodukt ihrer Produktionskette, dann ist die Grenze zwischen Herstellungs- und Anwendungsbereichsebene erreicht.

Im Bereich der Anwendungselemente gibt es so viele Abstraktionsebenen, wie aus der Sicht des Anwendungsbereiches sinnvoll sind. Auch hier gibt es verschiedene Kunden-Lieferanten-Verhältnisse. Der Composer ist im Regelfall ein Konstrukteur für Anwendungen, die als Variantenkonstruktion aus bestehenden Anwendungselementen komponiert werden. Das Gestaltungsprinzip der **Wiederverwendung durch Auswahl** ist dabei für Anwendungselemente eine Grundvoraussetzung. Für einen Composer darf keine Verpflichtung bestehen, Programmcode für das Zusammenspiel der Komponenten zu entwickeln. Denn die Personen, die diese Rolle in der Regel ausfüllen, sind Fachgebietsexperten, wohl mit einem unternehmensübergreifenden Problem- und Strategieverständnis, für Programmieraufgaben jedoch nicht geeignet. Anwendungselemente müssen so konzipiert und realisiert sein, dass sie bei korrekter Kombination eine funktionsfähige Anwendung ergeben (siehe Abschnitt 3.3.8).

---

[26] Diese Einteilung wird bei Oberon/F als „Creator“ und „Integrator“ bezeichnet [Fitié95].

In der Herstellungsebene ist die Scriptprogrammierung oder die Anpassung von Komponenten insofern unproblematisch, da Creator in der Regel ein sehr gutes Verständnis der Funktionsweise dieser Bauteile haben.

Nun gibt es, bedingt durch die unterschiedlichen Abstraktionsanforderungen des Anwendungsbereiches, nicht nur elementare Anwendungselemente, sondern auch Anwendungselemente-Baugruppen, die durch Komposition hergestellt werden. Baugruppen sind keine fixierten Lösungen, sie stellen optionale Zusammenbaupläne dar. Komplexe Baugruppen, die ganze Anwendungsbereiche repräsentieren, werden als Referenzmodelle, einfache Baugruppen als Mikro-Referenzmodelle bezeichnet. Während der Konstruktionsphase können Baugruppen wie Anwendungselemente behandelt werden. Ein Composer kann seine Sicht dadurch auf die Abstraktionsebenen begrenzen, die ihm bekannt sind.

Durch diese Aufteilung der Entwicklungsaufgaben wird auch das Kommunikationsproblem zwischen Anwendern und Entwicklern vermindert. Wie schon in Abschnitt 3.3.1 erwähnt, ist die aufgabenorientierte Gestaltung von Anwendungselementen ein wichtiger Schritt in die richtige Richtung. Durch die Einrichtung einer Kommunikationsbasis, den elementaren Anwendungselementen, ist die Verbindung der Denkweise der Creator, die mehr in Restriktionen und Bedingungen denken und den Anwendervertretern (Composer), die mehr in Zwecken und Inhalten (Aufgaben) denken, geschaffen. Elementare Anwendungselemente sind „Begriffe“ des Anwendungsbereiches, welche von den Vertretern beider Ebenen verwendet werden (müssen!).

Die folgende Grafik zeigt die Designrichtungen der unterschiedlichen Interessengruppen.

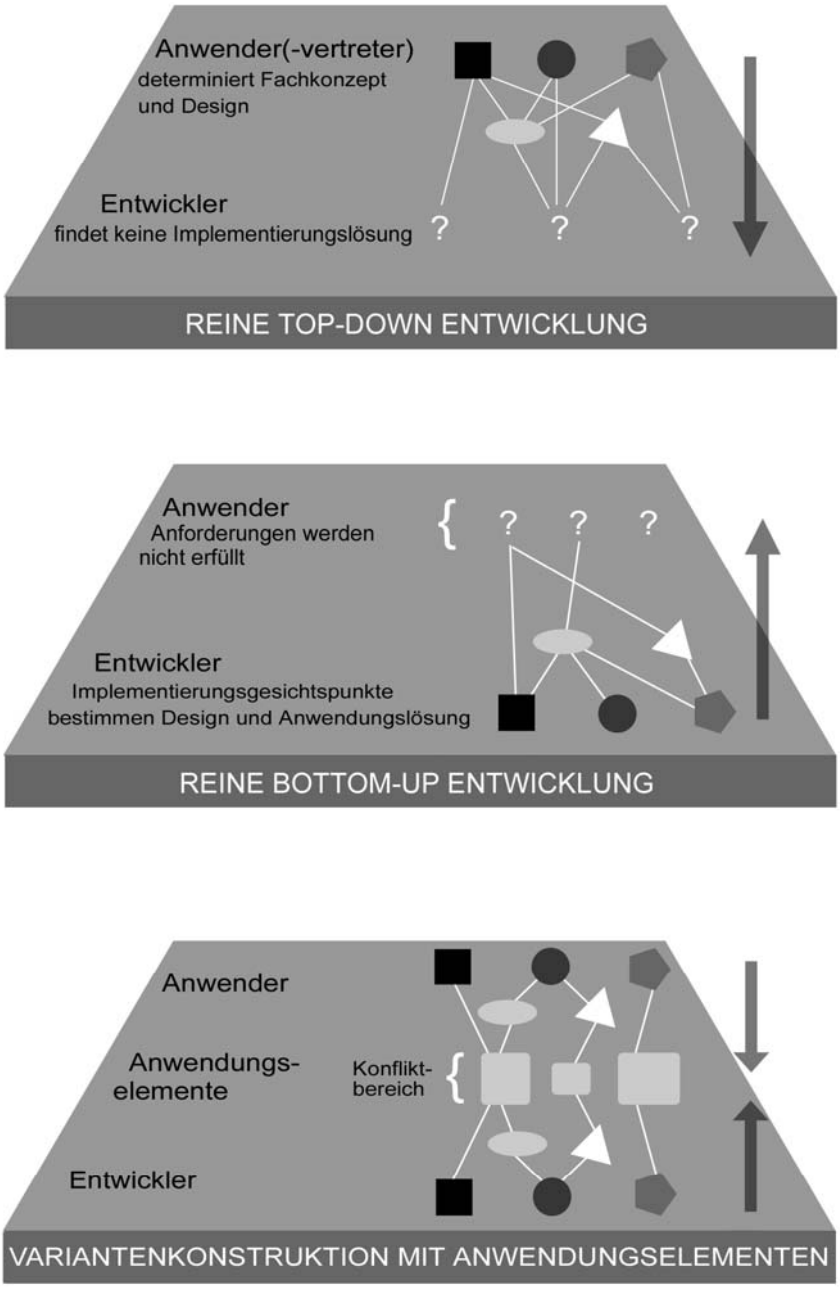


Abbildung 19: Anwendungselemente als Kommunikationsbasis

Der Konfliktbereich wird eingegrenzt auf die Definition und Verwendung von Anwendungselementen. Damit sind nicht alle Konflikte beseitigt, allerdings werden dadurch Kommunikationsgräben überbrückt.

Im Folgenden soll die Argumentation für das Gestaltungsprinzip der reinen Auswahl von Anwendungselementen für die Composer nachdrücklich unterstützt werden.

3.3.6 Wiederverwendung durch Auswahl versus Wiederverwendung durch Anpassung

Obwohl das so genannte „black-box-reuse“ den besten Wiederverwendungseffekt verspricht [Prieto-Diáz93, Goldberg/Rubin95], wird unmittelbar der Einsatz von Softwarekomponenten durch Modifikation und Anpassung als übliches Vorgehen bezeichnet. Bei [Pree97, 19f] unterstützt die „Erfahrung“ eines Frameworks, begründet durch häufige Wiederverwendung, einen Wechsel von white-box-Verwendung hin zu black-box-Verwendung. Dasselbe gilt für Softwarekomponenten, wobei die „black-boxes“ dann meist als parametrisierbare Komponenten mit guter Parameter-dokumentation realisiert werden.

Durch Vererbung und Polymorphismus stellt die Objektorientierung eine geeignete Entwicklungsmethodik für eine „Wiederverwendung durch Anpassung“ zur Verfügung.

Jedoch ist die Komponententechnologie mit ihrem Ziel, möglichst unabhängige (orthogonale) Bauteile zu verwenden, **nicht** dasselbe wie Objektorientierung. Denn Orthogonalität und Vererbung sind entgegengesetzte Anforderungen. Fertige Komponenten stehen zueinander nicht in Vererbungsbeziehungen. Sie sind eigenständig bzw. selbstständig. An dieser Stelle wird nicht die Herstellung von Komponenten betrachtet. Komponenten können durchaus objektorientiert oder mittels Schnittstellenvererbung hergestellt werden. Ihr Endzustand muss allerdings eine eigenständige Einheit darstellen, die mit ihrer Umwelt nur über Import- und Exportschnittstellen kommuniziert.

Die folgende Grafik zeigt eine Einteilung der Wiederverwendung entsprechend der Art, wie Komponenten in neuen Softwareprodukten eingesetzt werden können:

Abbildung 20: Wiederverwendungsarten [Lang98, 9]

Die ungeplante Wiederverwendung mit ihren Entscheidungskonflikten, Neuprogrammierung oder Adaption des Programmcodes, wird hier nicht näher untersucht. Damit Komponenten im Sourcecode ohne Veränderungen verwendet und dennoch in vorgedachten Bereichen durch die Benutzer an ihre Bedürfnisse flexibel angepasst werden können, haben sich Parameter als „Einstell-Schalter“ etabliert. Schalter können vorherbestimmte oder unbestimmte Zustände (Wertebereiche) annehmen. Diese Form der Wiederverwendung bedingt die Einsicht über interne Zusammenhänge und Funktionen der Komponenten und wird hier als „Wiederverwendung durch Einstellung“ bezeichnet.

Probleme bei der Parametrisierung treten in folgenden Ausprägungen auf:

- Ungeplante Schalterkombinationen erzeugen Konflikte innerhalb einer Komponente
- Schalterstellungen unterschiedlicher Komponenten, die jedoch in Beziehung zueinander stehen, erzeugen Konflikte zwischen Komponenten

Eine alternative Vorgehensweise zur Flexibilisierung von Komponenten ist die Variantenbildung. Für unterschiedliche Eigenschaften von Komponenten werden nicht Schalter zur Verfügung gestellt, sondern es wird eine um diese Eigenschaft differente

Komponente als Variante der Ursprungskomponente erstellt und eingesetzt. Die Verwendung unveränderter Komponenten, folglich so wie sie sind, wird hier „Wiederverwendung durch Auswahl“ genannt.

Die Unterschiede zwischen Wiederverwendung durch Einstellung und Wiederverwendung durch Auswahl werden im Folgenden anhand eines Beispiels verdeutlicht und in einer Gegenüberstellung nochmals ausführlich dargestellt.

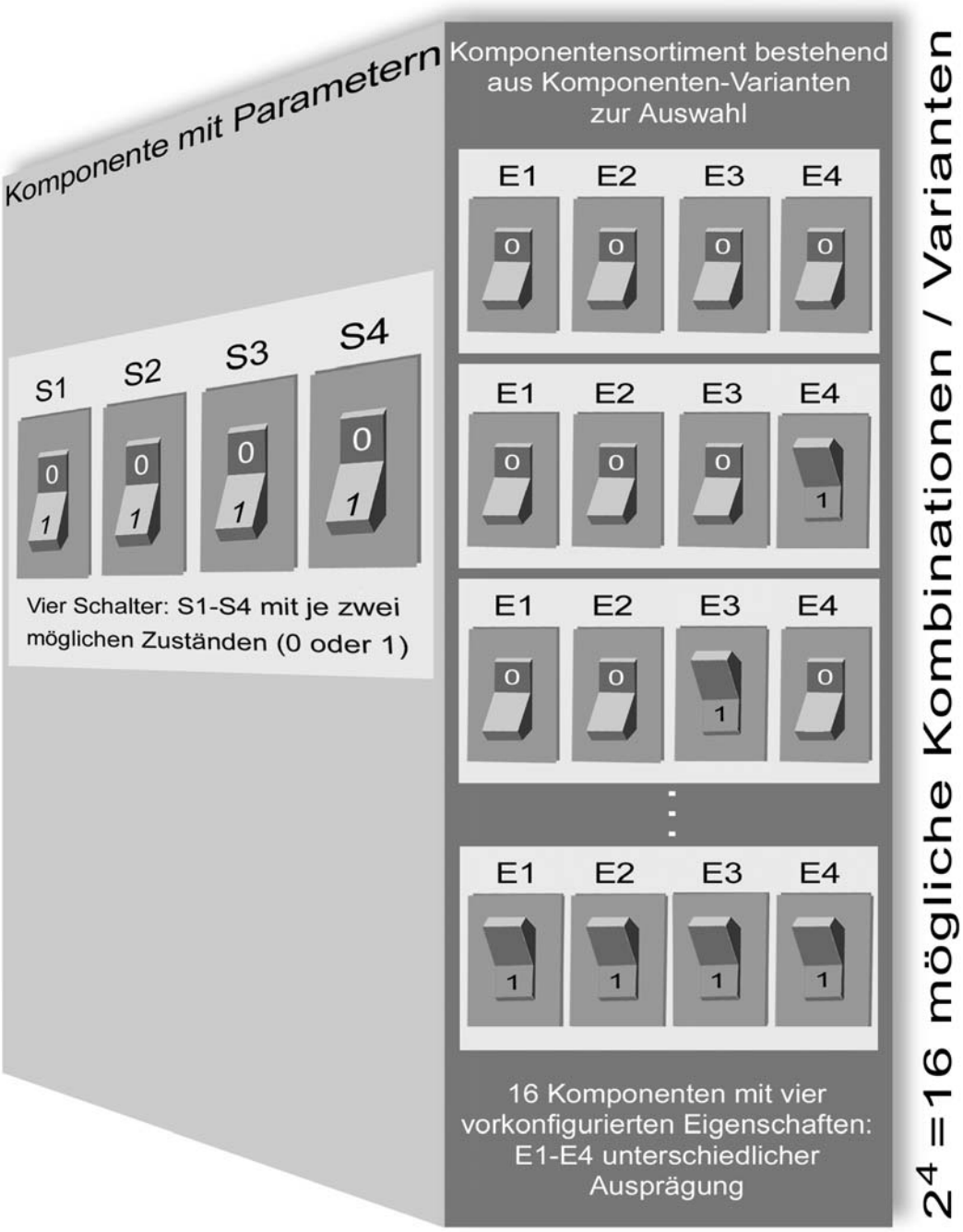


Abbildung 21: Parameter versus Variantenauswahl

Die Abbildung zeigt links eine Komponente, deren Parameter je zwei Werte annehmen können. Rechts wird ein Sortiment an Komponenten-Varianten dargestellt, die intern zwar ebenfalls über die Einstellung der Parameter „hergestellt“ wurden, deren „Eigenschaften“ jedoch definiert sind. Eine Variation kann nur über die Auswahl verschiedener Komponenten erfolgen. Die Herstellung von Varianten über interne Parameter ist nur eine Möglichkeit von vielen. Eine Parameter-Wertzuweisung wird in den folgenden Ausführungen wie eine Eigenschafts(ausprägungs-)zuweisung behandelt.

Für die weitere Betrachtung des Für und Wider von Parametrisierung und der unveränderten Verwendung von Komponenten im Auswahlverfahren ist der Aspekt der unterschiedlichen Abstraktionsebenen von Elementen und Baugruppen nicht zu vernachlässigen. Die Eigenschaften einer Baugruppe werden durch Auswahl und Komposition elementarer Komponenten definiert. Die nachfolgende Grafik veranschaulicht diesen Sachverhalt:

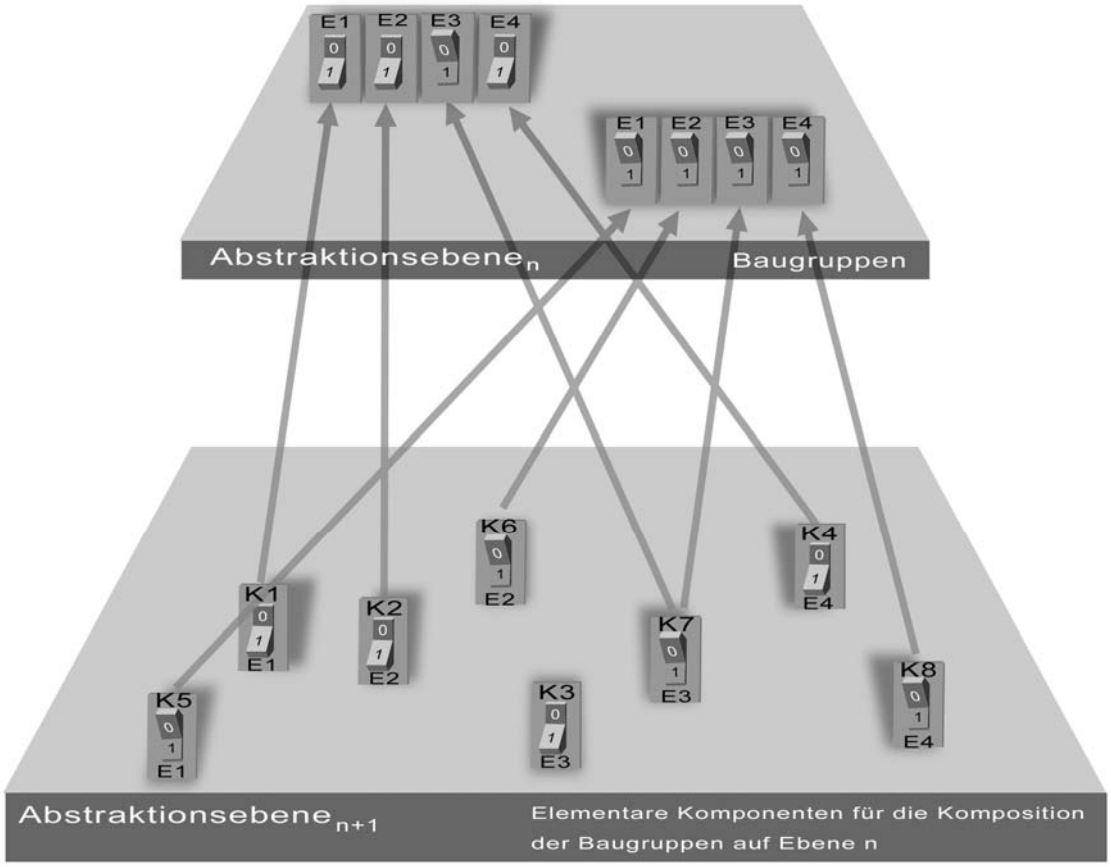


Abbildung 22: Erzeugung von Baugruppen

Für den Fall, dass die Parameterfestlegung der geforderten Variabilität an eine Komponente entspricht, ist es sinnvoll auch die Komponentisierung derselben genauso durchzuführen. Demnach unterscheidet sich die Komposition einzelner Elemente nicht prinzipiell von der Parametrisierung.

Es gibt dennoch wesentliche Unterschiede. Der Abstraktionsebenen-Wechsel für die Baugruppen-Komposition ist dabei zu berücksichtigen. Denn für einen Vergleich der beiden Prinzipien ist eine Betrachtung auf derselben Abstraktionsebene notwendig. Dann steht eine Komponente mit mehreren Schaltern einem Komponenten-Sortiment, bestehend aus mehreren Komponenten-Baugruppen mit genau definierten Eigenschaften, gegenüber.

Der vermeintliche Vorteil von parametergesteuerten Komponenten, dass nur die jeweiligen Schalter programmiert werden müssen und die Benutzer dann völlige Freiheit in der Wahl der Zustände und deren Kombinationen haben, existiert in Wirklichkeit nicht. Voraussetzung dafür wäre, dass die einzelnen Parameter vollständig orthogonal zueinander konzipiert und realisiert wären. Eine so starke Unabhängigkeit ist schon innerhalb einer Komponente sehr schwierig umzusetzen, zwischen den Komponenten eines integrierten Anwendungssystems nahezu unmöglich.

Die „Wiederverwendung durch Auswahl“ ist für den Bereich der Konstruktion von Anwendungssystemen aus Anwendungselementen der Parametrisierung eindeutig vorzuziehen. Denn auf der fachlichen Ebene des Anwendungsbereiches darf die Komponentenauswahl möglichst keine Konflikte mehr erzeugen und die Administration der Komponenten bezüglich Wartung und funktionaler Veränderung muss auf ein Komponentenverwaltungssystem übertragen werden können (siehe Kapitel 4).

Parametrisierung	Wiederverwendung durch Auswahl
Vorteile: Übersichtlichere Darstellung aller möglichen Zustände einer Komponente. Der Aufwand für die Vorkonfiguration der Komponenten entfällt.	**Vorteile:** Ein Kombinationstest für die Eigenschaften einer Baugruppe muss erfolgen. Die Verantwortung dafür liegt beim Hersteller der Baugruppe und nicht beim Benutzer. Für fertiggestellte Komponenten bzw. Baugruppen ist es einfacher einen Systemtest zur korrekten Ermittlung von semantischen Relationen (Wechselwirkungen zwischen Komponenten) durchzuführen. Mit der Komplexitätsproblematik zur Kombination von Eigenschaften einer Baugruppe muss sich ein Benutzer nur dann befassen, wenn er mit der Auswahl im Sortiment nicht einverstanden ist.
Nachteile: Treten Konflikte innerhalb von Komponenten auf, so ist es Aufgabe der Benutzer sie zu erkennen und evtl. zu beheben. Die Wechselwirkungen zwischen Komponenten werden durch die Wechselwirkungen zwischen einzelnen Parametern multipliziert. Semantische Relationen müssen für jede denkbare Schaltermöglichkeit verwaltet werden, obwohl einige Zustände keinen fachlichen Sinn ergeben. Der Benutzer der Komponente muss Entscheidungen über die sinnvolle Eigenschaft **aller** Parameter einer Komponente treffen.	**Nachteile:** Jede Eigenschaftskomposition muss ausdrücklich erstellt werden, auch wenn sie keine Konflikte beinhaltet. Es werden Komponenten hergestellt, die vielleicht nie zum Einsatz kommen. Kombinationen, die angefordert sind, wurden nicht erstellt.

Tabelle 5: Gegenüberstellung von Parametrisierung und Wiederverwendung durch Auswahl

Dem Nachteil der mangelhaften Übersichtlichkeit der Varianten bei der Wiederverwendung durch Auswahl muss durch ein strukturiertes Ordnungssystem und Regeln für die Verwaltung der Komponenten begegnet werden (siehe Abschnitt 3.4). Die Übersichtlichkeit wird weiter erhöht, indem nur die für die Auswahl relevanten Eigenschaften der

Anwendungselemente in einer ersten Informationsstufe angezeigt werden (explizite Eigenschaften).

Mit zunehmender Erfahrung in den Anwendungsbereichen und deren Sortimenten werden nicht eingesetzte Anwendungselemente erkannt und aussortiert. Nicht erfüllte Anforderungen an Eigenschaftskombinationen werden umgesetzt, wodurch ein möglicher Anwendungsstau abgebaut wird.

3.3.7 Anwendungselemente-Baukastensysteme

Zielsetzung eines Baukastensystems für Standardsoftware muss es sein, möglichst mit den vorhandenen Bausteinen (Anwendungselementen) auszukommen. Damit dieses Ziel erreicht werden kann, ist es wichtig, die Variabilitätsanforderungen möglicher Anwendungsfälle richtig vorherzusehen.

Jede Komponente kann in ihre Bestandteile zerlegt werden. Ist eine Partition (Gelenkbildung)[27] im Entwurf nicht umgesetzt, so ist dies eine Einschränkung der Variabilität des Baukastensystems und somit können nur mit geringerer Flexibilität Gesamtsysteme erzeugt werden. Umgekehrt, ist eine Komponente in Bestandteile zerlegt, so sind, auch wenn für jedes Element nur ein (Varianten-) Exemplar existiert, „latente“ Freiheitsgrade vorhanden, die bei Bedarf (Kundenanforderung) leicht durch Erzeugen weiterer Varianten genutzt werden können.

Eine Entscheidung bezüglich dieser Flexibilität ist ein Optimierungsproblem mit den Zielkonflikten zwischen kurzfristiger und langfristiger Wirtschaftlichkeit und somit Wettbewerbsfähigkeit eines Produktes bzw. einer Produktfamilie. Der Verzicht auf die Variabilität ist preiswerter und schneller zu realisieren („time to market“). Allerdings kann langfristig bei Veränderung der Kundenanforderungen, begründet durch Markt- und Geschäftsveränderungen, eine nachträgliche Flexibilisierung notwendig werden. So ein Vorhaben erzeugt dann unter Umständen ein Mehrfaches an Kosten. Es ist also wichtig, für die Tiefe der Partitionierung aktuelle und zukünftige Kundenanforderung gut zu kennen und vorherzusehen.

---

[27] Die hier angesprochene Partition kann man sich als Gelenk (joint) oder Verbindungsstelle vorstellen. Ist ein Gelenk realisiert, so kann an dieser Stelle des Systems die Variabilität ausgebaut werden.

In einem Baukastensystem für Anwendungssoftware werden die Anwendungselemente mit expliziten und impliziten Eigenschaften beschrieben. Die expliziten Eigenschaften dienen der Differenzierung von Anwendungselementen im Auswahlprozess für die Komposition. Es sind Eigenschaften, die der Aufgabensicht eines Anwendungsbereiches entstammen.

Implizite Eigenschaften werden in drei Fällen erstellt:

1. Es existieren Eigenschaften, die im Anwendungsbereich eindeutig erkannt und verstanden werden, die jedoch mangels Angebot an unterschiedlichen Anwendungselemente-Varianten nicht in der Menge der expliziten Eigenschaften aufgeführt werden. Man könnte sie als „latente“ explizite Eigenschaften bezeichnen, die bei Sortimentserweiterung aktiviert werden.
2. Wenn Eigenschaften für eine ausführliche Dokumentation den Anwendungselementen zugeordnet werden, damit Inhalt und Funktion besser verstanden werden.
3. Es gibt Eigenschaften, die eine Selektion des Bausteinbestandes im Baukastensystem durch außerhalb getroffene Vereinbarungen unterstützen sollen.

Dieser dritte Fall wird als Filterfähigkeit eines Baukastensystems bezeichnet. Es gibt so genannte Filtersysteme. Dies sind entweder Softwaremodule oder eigene Baukastensysteme außerhalb des Anwendungselemente-Baukastensystems, die für Systemanpassungen mit sehr einschneidenden Auswirkungen auf das Angebot eines Baukastensystems konzipiert sind (siehe Abschnitt 3.3.8). Die Einflüsse aus Abschnitt 3.3.2 sollten zum Teil als Filtersysteme realisiert werden.

Beispielsweise werden Stammdaten-Festlegungen vor der aufgabenorientierten Konstruktion des Anwendungssystems durchgeführt. Dadurch werden Anwendungselemente aus dem Sortiment genommen (gefiltert), die mit dieser festgelegten Stammdatenkonfiguration nicht kooperieren können. Ist die Stammdatenkonfiguration als Filtersystem realisiert, kann sie diese Filterung durchführen. Hat ein Unternehmen beispielsweise keine konfigurierbaren Produkte (z.B. Telefonanlagen) im Angebot, so sollten bei Abwahl dieser Eigenschaft im Filtersystem (Stammdaten-Verwaltung) auch

die Anwendungselemente, die sich auf konfigurierbare Produkte beziehen im Sortiment des Baukastensystems nicht mehr sichtbar sein.

Weitere Beispiele sind der Ländereinsatz, der bestimmte Vorschriften[28], Gesetze und die Anwendersprache regelt, oder Ergebnisse der Organisationsmodellierung, die Einfluss auf die Mandanten-, Geschäftsbereichs- oder Vertriebsbereichsfähigkeit der Anwendungselemente nehmen. Auswertungen (Berichte) erfordern ebenfalls Eigenschaften, die ganzen Gruppen von Anwendungselementen zugeordnet sein müssen.

An dieser Stelle wird nicht weiter auf die Eigenschaften von Stammdaten und Organisationsschemata zur differenzierten Ausprägung von Funktionen und Prozessen eingegangen [Thome/Hufgard96]. Die Differenzierung führt zu unterschiedlichen Anwendungsbauplänen je Organisationseinheit (z.B. Produktionsstandort) oder Stammdatengruppierung (z.B. Kundengruppe).

Filtersysteme können in unterschiedlichster Form auftreten; Es hängt völlig vom Anwendungsbereich des Baukastensystems ab. Auf jeden Fall ist eine Unterstützung von Filtersystemen eine unabdingbare Voraussetzung für die Akzeptanz der Anwendungselemente-Baukastensysteme im Konstruktionsprozess.

Als weitere Besonderheit eines Anwendungselemente-Baukastensystems gilt, dass Baugruppen genau gleich verwaltet werden wie elementare Bausteine. Nur so wird ein flexibler Wechsel zwischen verschiedenen Abstraktionsebenen beim Konstruktionsprozess ausreichend ermöglicht. Baugruppen werden folglich in der Konstruktionsumgebung eigenständig beschrieben, dokumentiert und identifiziert. Ein Zugriffssystem für die Auswahl der Bausteine muss dies entsprechend berücksichtigen.

Die aufgabenorientierte Elementarisierung eines Baukastensystems wird durch den Aufbau des Ordnungssystems (siehe Abschnitt 3.4) festgelegt. Das Ordnungssystem repräsentiert somit die Baukastenstruktur.

Als Folge einer Baukastensystematik können hier noch einige Richtlinien für die Gestaltung der Bausteine erstellt werden:

---

[28] Beispielsweise unterschiedliche kaufmännische Rundungsregeln (Deutschland – Schweiz).

- Mit abnehmender Einsatzhäufigkeit ist der Entwicklungsaufwand zu reduzieren [Borowski61, 11]. Das bedeutet, dass in Bausteine, die erwartungsgemäß häufig zum Einsatz kommen mehr Aufwand für die Verbesserung der Handhabbarkeit und der Performance investiert werden sollte als in andere.
- Auf der elementaren Ebene sollten sowenig Bausteine wie nötig entwickelt werden, die für den flexiblen Aufbau von vielen Gesamtsystemvarianten dennoch ausreichend sind (Orthogonalisierung).
- Zwischen Bausteinen sollten so wenig Schnittstellen wie möglich existieren [Richter84] (siehe auch Abschnitt 3.3.4.2).

Auf die Kommunikation zwischen Anwendungselementen in Baukastensystemen wird im nachfolgenden Abschnitt detailliert eingegangen. Kommunikationsanforderungen, bedingt durch die Bildung neuer Systemlösungen, werden hier kurz aufgeführt.

Neue Lösungen entstehen in einem Anwendungselemente-Baukastensystem durch folgende Möglichkeiten (Evolutionstheorie, siehe auch Abschnitt 3.3.1):

1. Herstellung neuer Varianten von Anwendungselementen innerhalb bestehender und neu gegründeter Sachmerkmalleisten.
2. Neukombination von komplexen Anwendungselementen (Baugruppenbildung).
3. Zerlegung eines elementaren Anwendungselementes in seine (möglichst) unabhängigen Bestandteile sowie Variation eines oder mehrerer Bestandteile und wiederholte Neukombination.

Für Neukombination gilt, dass die Schnittstellen innerhalb des Baukastensystems unberührt bleiben. Eine Variantenherstellung führt zu zwei alternativen Möglichkeiten bezüglich ihrer Schnittstellen:

a) Anwendungselemente, die völlig unabhängig von anderen Anwendungselementen sind.

b) Anwendungselemente, die mit anderen Anwendungselementen in besonderer Weise interagieren können bzw. ausdrücklich nicht interagieren dürfen.

Für a) ergibt sich daraus, dass ein Zusammenwirken von neuen, unabhängigen Anwendungselementen mit anderen bereits bestehenden Anwendungselementen im Gesamtsystem durch eine Integrationsbasis geleistet werden muss. Die Integrationsbasis muss für die neuen Anwendungselemente entweder neue Schnittstellen anbieten oder sie können mit bestehenden Schnittstellen arbeiten, die bisher nicht genutzt wurden oder jetzt in einer anderen Art interpretiert werden. Dieselben Schnittstellenmöglichkeiten gelten für b), jedoch betrifft es Schnittstellen zwischen den neuen und bereits bestehenden Anwendungselementen. Dabei müssen dann immer Beziehungen zwischen den betroffenen Anwendungselementen aufgebaut und verwaltet werden (siehe Abschnitt 3.4.4).

Einsatz von kundenindividuellen Bausteinen

Kundenindividuelle Bausteine bzw. auftragsspezifische Bausteine werden in Abbildung 10 als Nichtbausteine bezeichnet. Dies ist für Anwendungselemente unzutreffend, denn ein Anwendungselemente-Baukastensystem ist für den Einsatz projektindividueller Bausteine geradezu geeignet. Die zugrunde liegende Idee ist, dass Schnittstellen innerhalb des Baukastensystems genormt und veröffentlicht werden. Dadurch verpflichtet sich der Hersteller für eine längerfristige Stabilität. Für die Weiterentwicklung (Evolution) von Baukastensystemen wird es auf Herstellerseite eine Erweiterungsplanung geben. Werden Anwendungselemente-Varianten entsprechend den Richtlinien dieser Erweiterungsplanung erstellt, so macht es aus der Sicht des Baukastensystem-Herstellers keinen Unterschied, wer die Bausteine entwickelt hat – ein Kunde, ein Partner oder einer seiner eigenen Mitarbeiter. Nach entsprechender Zertifizierung (Qualtitätssicherung) und Regelung der Verwertungsrechte können diese Bausteine im Sortiment mit angeboten werden. Ein Markt für Systemerweiterungen ist ebenfalls denkbar. Dann muss das Baukastensystem jedoch eine herstellerübergreifende Normierung repräsentieren. Die Vorteile eines Komponentenmarktes sind in [Lang/Kalkmann97] dargestellt.

Kundenindividuelle Baugruppen werden in Projekten gebildet und können nach entsprechender Zertifizierung in das Ordnungssystem mit aufgenommen werden. Dadurch wird die „Erfahrung“ des Ordnungssystems permanent erhöht.

Für Baukastensysteme gibt es noch eine Voraussetzung. Sie müssen ein Bauprogramm (geschlossene Baukastensysteme) oder einen Baumusterplan (offene

Baukastensysteme) besitzen [Borowski61, 15ff]. Ein Bauprogramm ist eine Übersicht von allen möglichen und zugelassenen Kombinationen aller Bausteine eines Baukastensystems. Ein Baumusterplan ist quasi ein Bauprogramm-Ausschnitt, falls für alle möglichen und zugelassenen Kombinationen der Bausteine keine Übersicht mehr erstellt werden kann, weil die entstehenden Kombinationen im Vorfeld nicht bekannt sind.

Ein Baumusterplan stellt die Komposition eines Gesamtsystems beispielhaft dar. Er Entspricht einem Anwendungsbauplan für ein gedachtes Unternehmen. Das Ordnungssystem für Anwendungselemente entspricht im Gegensatz dazu, den Referenzplänen für viele mögliche Lösungen einer Branche bzw. eines Anwendungsbereiches. Das Ordnungssystem soll Anleitung und Unterstützung für die Erstellung von Anwendungsbauplänen für konkrete Unternehmen geben.

3.3.8 Kommunikation zwischen Anwendungselementen

Die ***Methode der semantischen Komposition*** ist unabhängig von der technologischen Realisierung ihrer Komponenten (Anwendungselemente). Dennoch sind ihre Entwurfsergebnisse in verschiedenen Komponentenkonzepten abbildbar. Wie diese Abbildung aussehen kann und welchen Ansprüchen die Komponentenkonzepte gerecht werden müssen, zeigen die folgenden Ausführungen.

Ergebnisse des Ordnungssystems für Anwendungselemente (siehe Abschnitt 3.4) werden hier schon für die Formulierung von Anforderungen an die Kommunikation zwischen Anwendungselementen verwendet, denn die Kommunikationslösungen sollten vor dem Aufbau eines Ordnungssystems erörtert werden, damit ihre technische Machbarkeit bekannt ist.

Es gibt drei Aktionsbereiche, die wichtig für die Variabilität der Anwendungssysteme sind und die sich bestimmend auf die Anwendungselemente-Kommunikation auswirken:

1. Kombination
2. Variation
3. Filterung

Sie sind nachfolgend in einer Übersicht dargestellt:

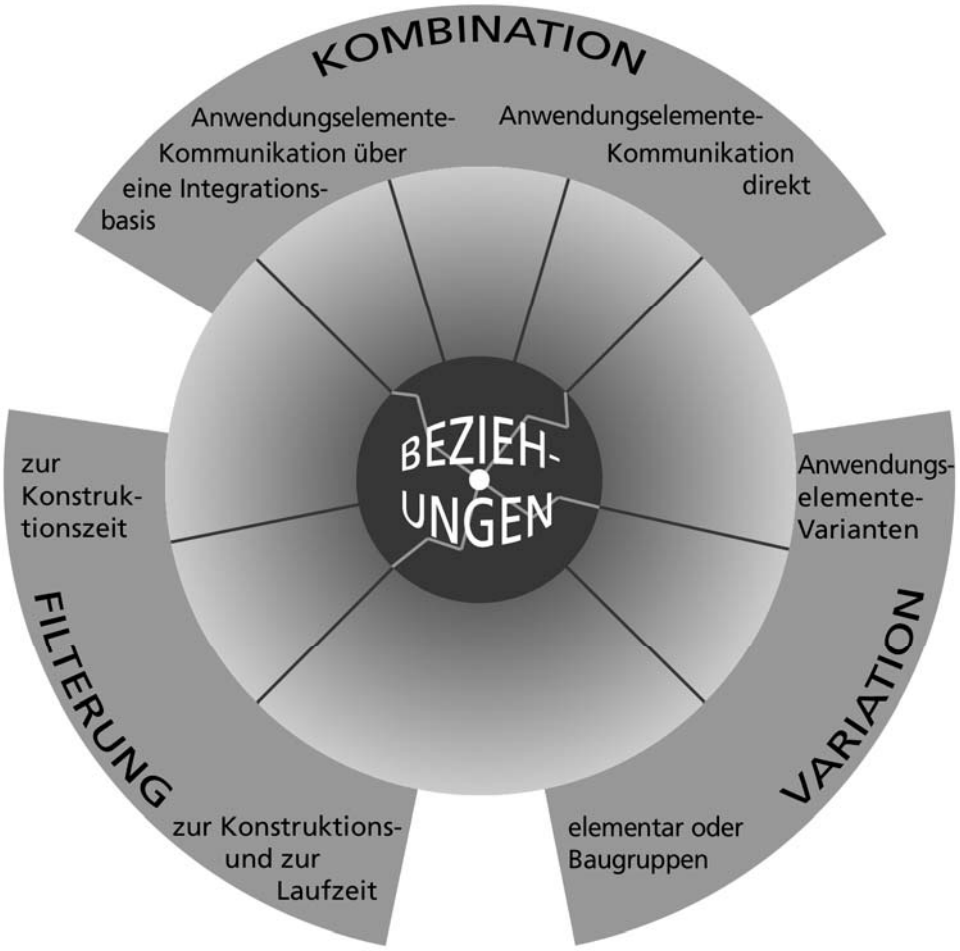


Abbildung 23: Aktionsbereiche für die Variabilität komponentenorientierter Anwendungssysteme

Der Einfluss der „Filterung“ muss in Bezug auf die Kommunikation zwischen Anwendungselementen und Filtersystemen betrachtet werden. Im Grunde stellt die Filterung eine zusammenfassende Eigenschaftsberücksichtigung vieler Anwendungs-elemente dar. Dadurch wird das Auswahlsortiment zur Konstruktionszeit vorselektiert („gefiltert“), oder es werden zur Laufzeit geeignete Anwendungselemente-Baureihen verwendet.

Für die Kommunikation der Anwendungselemente mit Workflow-/Prozess-Management-Systemen, Geschäftsregelverarbeitungs-Systemen oder Berechtigungs-verwaltungs-Systemen bzw. für die integrative Handhabung der Organisations-

festlegungen (Filtersystem für die Organisationsmodellierung) werden standardisierte Schnittstellen vorausgesetzt.

Für die Geschäftsregelverarbeitung sind zwei verschiedene Möglichkeiten zu berücksichtigen:

1. Ablaufbedingte Geschäftsregeln (z.B. Beschaffungsanträge über 2000,- EUR müssen von einem Abteilungsleiter geprüft werden...) können in Workflow-Management-Systemen für die Ablaufsteuerung verarbeitet werden.

2. Es gibt darüber hinaus noch Geschäftsregeln, die an neuralgischen Punkten im Anwendungssystem bereitgestellt werden müssen. Beispielsweise können dies Prüfungen bezüglich der Kreditwürdigkeit von Kunden sein. Geschäftsregeln dieser Art sind direkt mit der Ausführung von Anwendungselementen zu prüfen. Eine Schnittstelle ist dafür vorzusehen.

Hierbei wird deutlich, dass die Geschäftsregelverarbeitung direkt Einfluss auf die Anwendungselemente-Auswahl nehmen kann. Wird ein Anwendungselement gewählt, welches keine Regelprüfung vorsieht, so entsteht bei der Regeldefinition ein Konflikt.

Die Filterung hat nur insofern Einfluss auf Schnittstellen, als dass diese an den geforderten Stellen vorhanden sein müssen.

Für die Betrachtung von Kommunikationskonflikten zwischen Anwendungselementen und Integrationsbasis ist im Wesentlichen also die „Kombination“ und die „Variation“ der Anwendungselemente von Interesse. Kombination und Variation sind nicht beliebig möglich, denn es existieren Beziehungen (Abhängigkeiten) zwischen Anwendungselementen. Diese Beziehungen können sowohl fachlicher als auch technischer Art sein.

Fachliche Abhängigkeiten zwischen Anwendungselementen sollten „enger“ sein als ihre technischen Abhängigkeiten, andernfalls müssen auf der Anwendungsbereichsebene im Auswahlvorgang Anwendungselemente hinzugenommen werden, für die es betriebswirtschaftlich keinen Grund zur Auswahl gibt. Das bedeutet für die Realisierung, dass die Implementierungsvorschriften von Anwendungselementen

weniger oder höchstens dieselben Beziehungen besitzen dürfen, wie ihre betriebswirtschaftliche Verwendung (Inhalt).

Aufgrund dieser Forderung ist es sinnvoll, die inhaltlichen (semantischen) Beziehungen zwischen Anwendungselementen getrennt von ihren physikalischen Schnittstellen und deren „Erfüllung“ zu verwalten. Wobei implementierungstechnisch bedingte Abhängigkeiten neben den semantischen Beziehungen gleichbedeutend verwaltet werden müssen (siehe Abschnitt 3.4.4). Bevor nun die Anforderungen der Kombination und der Variation von Anwendungselementen an ihre Kommunikationsschnittstellen genau untersucht werden, sind erst noch die Bedingungen für die Kommunikation von Komponenten im Allgemeinen zu klären.

- Komponenten dürfen nur über ihre Schnittstellen angesprochen werden.
- Jeder Kommunikationseingang (Import/Input) und jeder Kommunikationsausgang (Export/Output) muss erkennbar sein.

Diese Bedingungen werden von der Entwicklungsrichtung „Programmieren im Großen“ [Börstler94] oder von der Software Architekturlehre [Shaw/Garlan96, Ning et.al. 94] ausdrücklich gefordert. Damit wird die Moduleigenschaft der Komponenten deutlich sichtbar [Hruschka86]. Cherrys „Stimulus-Response-Machine“ [Cherry91], von Prins aufgegriffen und in Bezug auf Business-Objekte dargestellt, lässt sich in seiner plakativen Art auch auf den Komponentenbegriff, wie er hier verwendet wird, anwenden. Eine Komponente wird demzufolge über ihre Inputschnittstelle angeregt, etwas zu tun und sollte mit einer Antwort reagieren [Prins96, 96ff].

Das Stimulus-Response-Modell gilt sowohl für die Kommunikation zwischen Anwendungselementen als auch für die Kommunikation zwischen Anwendungselementen und ihrer Integrationsbasis. Hinzu kommt, dass es auch für eine Interaktion mit so genannten Steuerungseinheiten gilt. Es können drei Arten von Steuerungseinheiten unterschieden werden:

1. Ereignissteuerung durch Benutzerinteraktion
2. Vorgangssteuerung durch ein Workflow-/Prozess-Management-System (inkl. Geschäftsregelverarbeitung)
3. Management verteilter Anwendungskomponenten

Zu 1.: Der Anwender und sein Interagieren mit den Anwendungselementen ist ein wesentliches Element für die Gestaltung von betriebswirtschaftlichen Anwendungssystemen. In der ***Methode der semantischen Komposition*** ist der „benutzerspezifische Arbeitsplatz“ als ein Methodenbaustein für die Komposition von Anwendungselementen für eine bestimmte Mitarbeiter-Rolle definiert (siehe Abschnitt 4.2). Die Aktivierung von Anwendungselementen, die ein Anwender direkt anspricht, wird an eine „Menü-Komponente“ delegiert. Diese Menü-Komponente steuert den wahlfreien Zugriff der Anwender auf bestimmte Anwendungselemente. Wie eine Menü-Komponente aussehen kann, ist abhängig von der eingesetzten Technologie. So kann sie beispielsweise durch eine HTML-Struktur realisiert werden: Dies ist der Fall, wenn die Anwendung in einem Internet-Browser abläuft. Im Projekt ***TKMS*** (siehe Kapitel 4) wurde es so realisiert.

Zu 2.: Die Vorteile einer Separierung der Ablaufsteuerung von den Anwendungselementen wurde schon mehrfach erwähnt. Es ist wichtig zu verdeutlichen, dass durch die Auslagerung der Ablaufsteuerung eine Veränderung des Ablaufplanes auch eine Veränderung des Stimulus-Response-Modells nach sich zieht. Sofern die Reihenfolge der Bearbeitung für Anwendungselemente keine Probleme bereitet, müssen keine besonderen Maßnahmen eingeleitet werden. Ist dies jedoch der Fall, so muss der Ablaufplan „filternd“ auf die Auswahlmöglichkeiten der Anwendungselemente innerhalb des Baukastensystems wirken. Bei Anpassungen eines produktiven Systems müssen Konflikte in der Konfiguration aufgedeckt und durch Auswahl aus dem Baukastenbestand und anschließender Komposition gelöst werden.

Zu 3.: Das Management verteilter Anwendungskomponenten [Zimmermann93] mit seinen Lösungsalternativen ist der Herstellungsebene zuzuordnen und wird hier nicht weiter diskutiert.

3.3.8.1 Anforderungen der Kombination und Variation von Anwendungs-elementen und Baugruppen an ihre Schnittstellengestaltung

Eine Betrachtung der Intention des Komponentenbaus hilft, die Schnittstellen-anforderungen zu verdeutlichen. Werden Softwaresysteme nur komponentisiert, damit eine Arbeitsteilung in der Entwicklung bzw. eine verbesserte Versionierung einzelner Module erreicht wird, so stellt die Schnittstelle zwischen diesen Komponenten eine Vereinbarung dar, deren Schnittstellen-Zugriffe „hard-wired“ realisiert werden können (Design by Contract) [Meyer96, 18ff]. D.h. der Schnittstellen-Zugriff zur Laufzeit adressiert eine zur Entwicklungszeit bereits bekannte Implementierung.

Erfolgt im Gegensatz dazu die Komponentisierung mit der Zielsetzung, möglichst flexible Kombinierbarkeit und möglichst hohe Austauschbarkeit zu erreichen, so liegt eine völlig andere Situation vor. „Design for Component“ [Fähnrich et.al. 97] kann nicht durch eine fest verdrahtete Kommunikation unterstützt werden. Die Schnittstellen stellen zwar auch Vereinbarungen bzw. Standards dar, allerdings muss ihre Verwendung so flexibel gehandhabt werden können, wie der Einsatz der zugehörigen Komponenten es erfordert. Das bedeutet, dass über eine Indirektion die Auswahl der Implementierung dynamisch erfolgen kann – entweder bei der Installation oder zur Laufzeit. Die Softwarekomponente für diese dynamische Verbindungserstellung wird „Dispatcher“ – Versender – genannt.

Die Forschung im Bereich „Module Interconnection Languages“ (MIL) hat schon frühzeitig eine Einteilung der Schnittstellen (Ressourcen) in **anfordernde** und **liefernde** Schnittstellen vollzogen [DeRemer/Kron76, Prieto-Diáz/Neighbors86].

Diese Erkenntnis nutzend, definiert die ***Methode der semantischen Komposition*** „Kunden-Schnittstellen“ und „Lieferanten-Schnittstellen“.

Zur Verdeutlichung der Verwendung der zwei Arten von Schnittstellen werden einige Eigenschaften aufgelistet:

- Ein Methodenaufruf erfolgt immer von der Kunden-Schnittstelle zur Lieferanten-Schnittstelle.
- Lieferanten-Schnittstellen benötigen keine Kunden-Schnittstellen. Es können zur Laufzeit jedoch viele Kunden-Schnittstellen dieselbe Lieferantenschnittstelle nutzen[29].
- Jede Kunden-Schnittstelle muss von genau einer Lieferanten-Schnittstelle bedient werden. Gibt es keine Lieferanten-Schnittstelle so entsteht ein „Kunden-Konflikt“. Gibt es mehrere gleichnamige Lieferanten-Schnittstellen, so entsteht ein „Lieferanten-Konflikt“.

Die folgenden zwei Grafiken (Abbildung 24 und Abbildung 25) stellen eine Übersicht der Kommunikationsmöglichkeiten zwischen Kunden- und Lieferanten-Schnittstellen dar.

[29] Regeln für die Lösung des Konkurrenzproblems werden hier nicht erläutert. Sie sind jedoch für die Implementierung durchaus relevant.

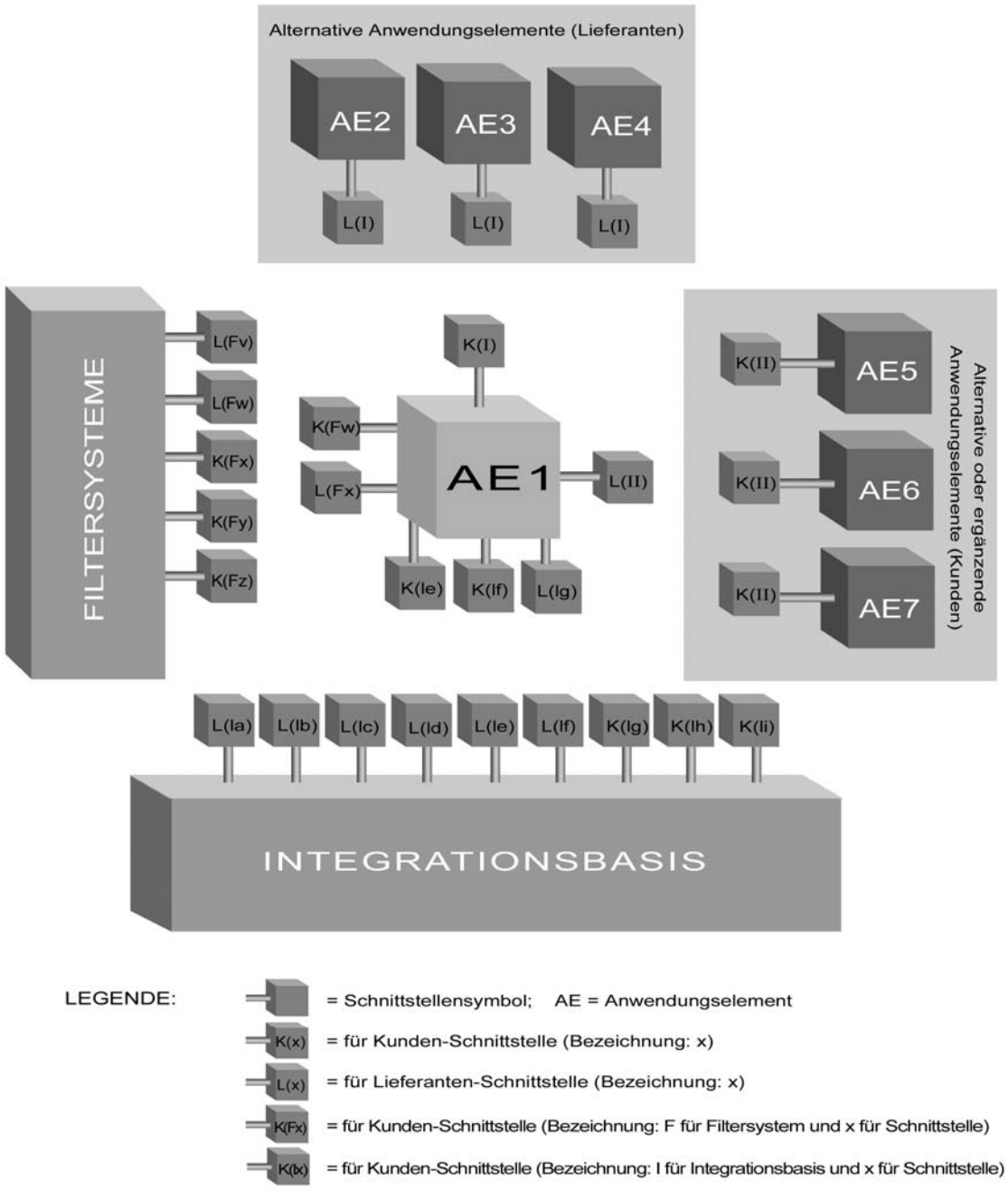


Abbildung 24: Kommunikationsmöglichkeiten des Anwendungselementes AE1

Anwendungselement 1 wird hier mit seinen Schnittstellen hin zur Integrationsbasis, zu Filtersystemen und zu verschiedenen Anwendungselementen im System dargestellt.

Kommunikation mit der Integrationsbasis

Die Integrationsbasis besitzt viele Kunden- und Lieferantenschnittstellen. Sie kann bezüglich ihrer Schnittstellen als sehr mächtiges globales Anwendungselement gesehen werden. Dadurch haben viele Anwendungselemente Kommunikationsabhängigkeiten zur Integrationsbasis. Es ist nicht notwendig, dass alle Lieferanten-Schnittstellen in einer Kundenkonfiguration genutzt werden (sogar eher unwahrscheinlich!). Für den Einsatz ihrer Kunden-Schnittstellen müssen die notwendigen Lieferanten-Schnittstellen in einem Anwendungssystem vorhanden sein. Um auch hierbei Flexibilität zu sichern, sollte eine Integrationsbasis selbst auch konfigurierbar sein. Ist dies der Fall, können ihre Kunden-Schnittstellen, je nach Anforderung, deaktiviert werden.

Kommunikation mit Filtersystemen

Filtersysteme sind ebenfalls als globale Anwendungselemente zu sehen[30]. Sie erfüllen zur Laufzeit zwei Aufgaben. Erstens können sie die Aufrufe der Anwendungselemente-Varianten und -Baureihen steuern, die nach der Komposition noch dynamisch zur Auswahl stehen und zweitens dient ihr globaler Servicecharakter dazu so genannte „Querschneidende Module“ [Ostermann04] mit durchgreifender Bedeutung abzubilden[31]. Aufgrund ihrer Querschnittsfunktion werden sie von sehr vielen Anwendungselementen aufgerufen. Auch Filtersysteme sollten konfigurierbar sein, damit die Anwendungselementeauswahl (zur Konstruktionszeit und zur Laufzeit) flexibel gestaltet werden kann.

[30] Es werden hier nicht die Filtersysteme betrachtet, die ausschließlich zur Konstruktionszeit zur Ausfilterung von vielen Anwendungselementen mit denselben Eigenschaften, wie z.B. Ländereinsatz, verwendet werden.

[31] „Querschneidende Module“ ist ein Design-Prinzip, welches immer wieder auftretende Verwendung von Verarbeitungsservices (Aspekte) zentralisiert, anstatt sie in einer Modul-Hierarchie über viele Module zu verteilen. Querschneidende Module kommen nicht nur in den Filtersystemen vor, jedoch sind die Filtersystem-Services typischerweise von dieser Art.

Kommunikation mit Anwendungselemente-Lieferanten

Die Forderung, dass zu jeder Kunden-Schnittstelle eine Lieferanten-Schnittstelle existieren muss, macht die Abhängigkeitsbeziehungen zwischen den jeweiligen Anwendungselementen deutlich.

Das Anwendungselement (AE1) mit der Schnittstelle [K(I)] benötigt ein Anwendungselement mit einer Schnittstelle [L(I)]. Es darf jedoch nur genau eine Schnittstelle [L(I)] im Anwendungssystem existieren, sonst ist zum Zeitpunkt eines Methodenaufrufs die Ziel-Schnittstelle nicht eindeutig. Folglich sind die Anwendungselemente AE2, AE3 und AE4 **alternativ** zu verwenden.

Wie schon erwähnt, sollten die technischen Abhängigkeiten höchstens gleich groß wie die fachlichen Abhängigkeiten sein. Es ist also notwendige Bedingung, dass zwischen den Anwendungselementen AE1 und (AE2 oder AE3 oder AE4) eine Implikationsbeziehung besteht (siehe Abschnitt 3.4.4). Zwischen den Anwendungselementen AE2, AE3, und AE4 muss eine Äquivalenzbeziehung oder Austauschbarkeit bestehen.

Kommunikation mit Anwendungselemente-Kunden

Die Lieferanten-Schnittstellen von AE1 können aus technischer Sicht unbenutzt bleiben. Inhaltlich bedeutet dies, dass hier Fähigkeiten eines Anwendungselementes implementiert wurden, die in einer speziellen Kundenkonfiguration nicht benötigt werden. Es ist möglich, der Schnittstelle [L(II)] eines oder mehrere Anwendungselemente mit der Schnittstelle [K(II)] gegenüberzustellen. Die Anwendungselemente AE5, AE6 und AE7 können dabei in beliebiger Kombination ergänzend oder alternativ mit AE1 zusammen eingesetzt werden.

Wird kurz der Blickwinkel verändert und die Situation aus der Sicht von AE6 betrachtet, so wird leicht ein Konflikt zwischen Anwenderwünschen deutlich. Gesetzt den Fall, es gäbe für AE1 eine Variante AE8, die ebenfalls [L(II)] zur Verfügung stellt, die jedoch während des Konstruktionsprozesses zusammen mit AE5 bewusst negiert wurde, so hat der Composer, der in einer anderen Aufgabenstellung für die Kommunikation mit AE6 ein Anwendungselement mit [L(II)] sucht, keine Wahlfreiheit mehr in Bezug auf AE8, mit anderen Worten, er kann AE8 nicht mehr auswählen, andernfalls würde ein Lieferanten-Konflikt auftreten. Der Composer muss sich entweder kompromissbereit

erklären und AE1 als zulässige, jedoch nicht als beste, Lösung akzeptieren, oder er muss sich mit den Entscheidungen der zuvor komponierten Lösung auseinandersetzen.

Die folgende Grafik zeigt die Kommunikationsmöglichkeiten von Anwendungselementen in einer Baugruppe und ihre potenziellen Kommunikationskonflikte.

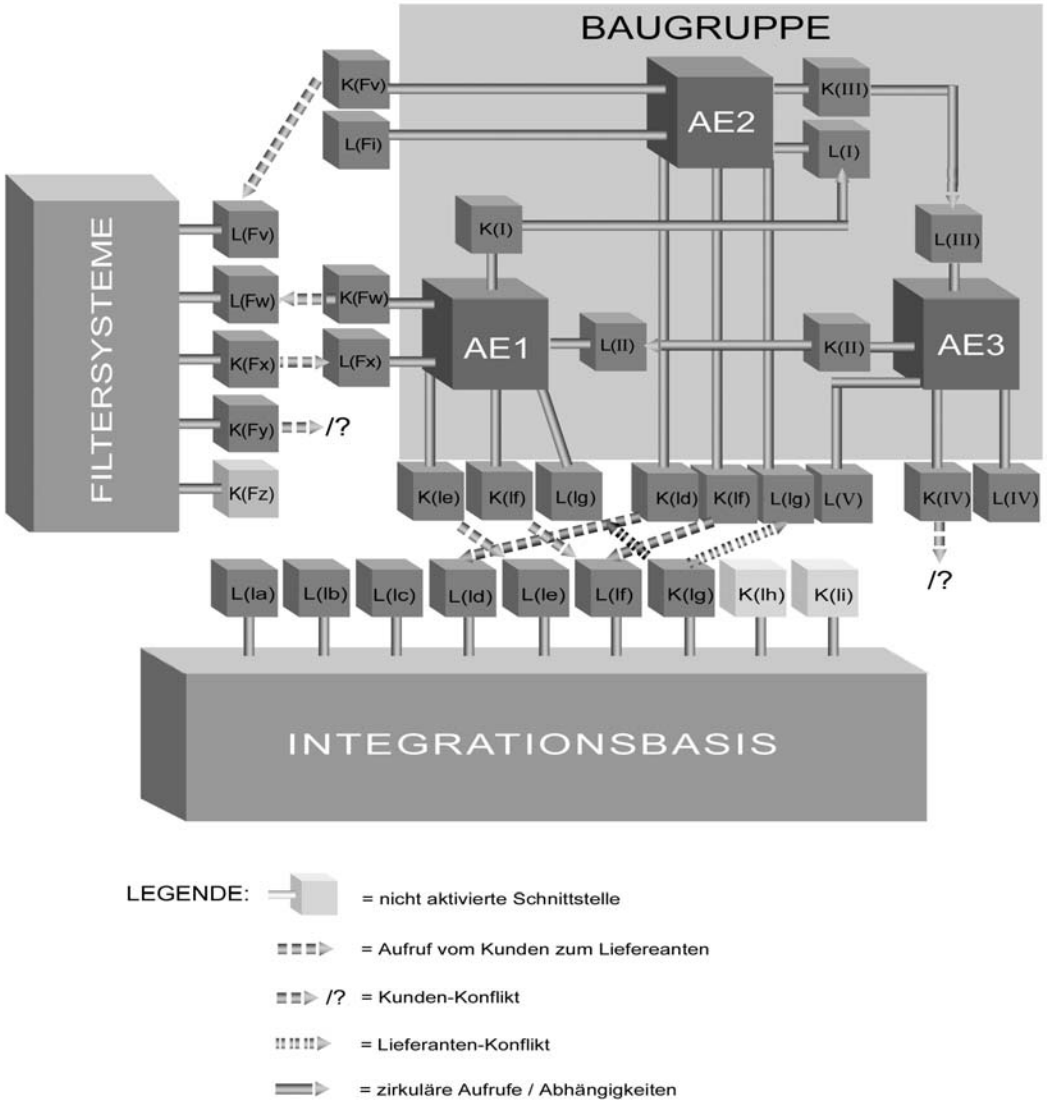


Abbildung 25: Baugruppen-Kommunikation und Kommunikationskonflikte

Eine Baugruppe ist eine Komposition von Anwendungselementen für einen bestimmten Zweck. Baugruppen sind im Baukastensystem bereits vorhanden (Referenzmodelle), oder sie werden in einem Konstruktionsprojekt gebildet. In jedem Fall repräsentieren Baugruppen eine bestimmte Abstraktionsebene. Steigt man auf bis zur Abstraktionsebene des Anwendungssystems, so kann die Zusammenstellung aller Lösungselemente als eine Baugruppe, nämlich die Anwendung, gesehen werden.

Die ***Methode der semantischen Komposition*** sieht eine Baugruppe zumindest auf der konzeptionellen Ebene als eigenständige Einheit (Kompositionsabstraktion [Börstler94, 34ff]). Auf dieser Ebene wird sie als selbstständiges Konstruktionselement behandelt mit eigener Beschreibung, Eigenschaftsstruktur und Einordnung in das Ordnungssystem (siehe Abschnitt 3.4). Da jedoch eine Technologieunabhängigkeit hinsichtlich der Implementierung angestrebt wurde, ist die Realisierungsform der Baugruppe nicht vorgeschrieben.

Eine Baugruppe kann nun als eine neue Komponente, die ihren Inhalt vollständig verbirgt, implementiert werden. Diese Form wird von der ***Methode der semantischen Komposition*** derzeit jedoch nicht vorgeschlagen. Sie hat zwar Vorteile bezüglich der oben aufgeführten Anwenderwunsch-Konflikte, allerdings wird sie nicht von jeder Realisierungstechnologie unterstützt. Zudem ist der Wartungsaufwand und die Benutzerzufriedenheit bei umfangreichen Systemen, die zeitgleich mehrere Versionen derselben Anwendungselemente einsetzen, als kritisch einzustufen.

Die andere Form der Realisierung sieht innerhalb eines Anwendungssystems (von Verteilungsaspekten wird hierbei abgesehen) die Singularität von Anwendungselementen vor. Das bedeutet, dass jede Baugruppe, die ein bestimmtes Anwendungselement beinhaltet, zur Laufzeit auch auf dasselbe Anwendungselement referiert. Dadurch wird die Wartung der Systeme vereinfacht. Eine Lösung des Anwenderkonfliktes kann dennoch angeboten werden. Ähnlich der Namensraum-Partitionierung in Java [Aitken96] kann für die Komposition von Anwendungselementen ein „Methodenaufruf-Gültigkeitsbereich“ definiert werden. Die Zuordnung von Schnittstellen zur Konstruktionszeit gilt dann nur für diesen Bereich. Der „benutzerspezifische Arbeitsplatz“, Geschäftsprozesse oder Unternehmensbereiche bieten sich für eine Namensraum-Segmentierung an. Die Schnittstellen-Kommunikation bleibt trotzdem eindeutig, obwohl im gesamten Anwendungssystem mehrere verschiedene Anwendungselemente mit derselben Lieferanten-Schnittstelle existieren.

Schnittstellen einer Baugruppe nach außen sind alle nicht innerhalb einer Baugruppe „bediente“ Schnittstellen. Dies können sowohl Kunden- und Lieferanten-Schnittstellen hin zu Anwendungselementen als auch zu Filtersystemen und zur Integrationsbasis sein.

In der oben aufgeführten Grafik werden die Kunden- und Lieferanten-Konflikte gezeigt. Außerdem wird noch auf eine Situation hingewiesen, die vermieden werden sollte: zirkuläre Aufrufe bzw. Abhängigkeiten.

Kompositionshandlungen, sowohl während der Systemeinführung als auch bei Änderungen im laufenden Betrieb, lassen sich in der Regel nicht auf einzelne Komponenten begrenzen. Häufig existieren verkettete Abhängigkeiten zwischen Anwendungselementen, so dass im Auswahlverfahren der Konstruktion die Wahl eines Anwendungselementes sofort die Wahl eines weiteren Anwendungselementes (oder eines seiner Varianten) nach sich zieht. Mit dieser Problematik muss ein System zur Unterstützung der Komposition zurechtkommen. Sind jedoch zirkuläre Abhängigkeiten vorhanden, so reduziert sich die Flexibilität eines Baukastensystems erheblich und es kann nicht mehr adäquat eingesetzt werden.

> Die **Reduktion von Abhängigkeiten** im Baukastensystem ist ein permanentes Modellierungsziel!

Fowler und Scott [Fowler/Scott97, 113ff] geben in Bezug auf die Anwendung der Unified Modeling Language (UML) einige sinnvolle Hinweise zur Vermeidung von „Dependencies“. Übertragen in die UML entsprechen Baugruppen den Packages, und die Integrationsbasis sowie die Filtersysteme stellen globale Packages dar. In Abschnitt 3.3.4.2 „Integrationsaspekt“ wurden ebenfalls schon Hinweise auf die Reduktion der Bindung gegeben, die in der Anwendungselemente-Modellierung ihre Anwendung finden sollten.

Die Austauschbarkeit von Anwendungselementen lässt sich unter Berücksichtigung ihrer Schnittstellen folgendermaßen ausdrücken.

Definition:

> **Austauschbarkeit** ist gegeben, wenn eine Komponente M durch eine andere Komponente M‘ ersetzbar ist, wobei M‘ mindestens die Dienste von

> M zur Verfügung stellt und höchstens die Dienste von M in Anspruch nimmt [vgl. Börstler94, 11].

Übertragen auf die hier vorgestellten Schnittstellen bedeutet dies, dass ein Anwendungselement X durch genau ein anderes Anwendungselement Y ersetzt werden kann, wenn Y mindestens die (in Anspruch genommenen) Lieferanten-Schnittstellen von X anbietet und höchstens die Kunden-Schnittstellen von X besitzt. Ausnahmen sind dabei die Schnittstellen hin zur Integrationsbasis. Wobei ein Verzicht auf Lieferanten-Schnittstellen dann möglich ist, wenn bei Konfigurationsänderung der Integrationsbasis die Deaktivierung von Kunden-Schnittstellen erfolgt. Zudem kann Y in Richtung Integrationsbasis beliebig mehr Kunden-Schnittstellen als X besitzen.

3.3.8.2 Realisierungsmöglichkeiten der Komponenten-Kommunikation

Die Realisierung der Kommunikationsbedürfnisse von Anwendungselementen ist weniger eine Frage der technischen Machbarkeit als eine Frage der am besten geeigneten Lösung.

Die folgende Aufstellung fasst die Kernforderungen an eine Realisierung der Komponenten-Kommunikation unter Berücksichtigung der Bedürfnisse für eine Komposition rein durch Auswahl, wie sie für die ***Methode der semantischen Komposition*** gefordert wird, zusammen:

- Explikation der Kunden- und Lieferanten-Schnittstellen
- Realisierung des Parameterpolymorphismus
- Realisierung eines durch Komposition aktivierten Verbindungsmechanismus

Im Prinzip kann mit jeder Programmiersprache, die eine eigene Unterstützung für Schnittstellen (Interfaces) anbietet (beispielsweise C++, Java, Ada), eine derartige Kommunikationslösung implementiert werden.

Die Vereinbarungen, die dabei getroffen und eingehalten werden müssen, sind jedoch sehr umfangreich. Um diese Vereinbarungen zu konsolidieren und auf eine konsistente Basis zu stellen, wurden eigene „Sprachen“ für die Komponenten-Kommunikation entwickelt. Die Umsetzung dieser Komponenten-Kommunikations-Sprachen in

Implementierungssprachen erfolgt entweder direkt oder mit Unterstützung durch Konverter.

Vertreter dieser Sprachen sind die Module Interconnection Languages aus dem Bereich „Programmieren im Großen“ [DeRemer/Kron76, Prieto-Diáz/Neighbors86, Börstler94] und der Software Architektur-Konzepte [Shaw/Garlan96]. Neben diesen Ansätzen, die zum Teil sehr auf die Konzeption fokussiert sind, haben praxisnahe Projekte die Umsetzung zumindest auf Prototypenebene nachgewiesen. Beispiele hierfür sind das ESPRIT Projekt 2080 „REX – Reconfigurable & Extensible Parallel and Distributed Systems“ [REX96], welches in Zusammenarbeit mit der Stollman GmbH, der Siemens AG und der Gesellschaft für Mathematik und Datenverarbeitung (GMD) durchgeführt wurde. Bei Andersen Consulting wurde ebenfalls ein Komponenten-Konfigurationssystem entwickelt [Ning et.al. 94], welches als ein architekturgetriebener, anwendungsspezifischer und komponentenbasierter (ABC) Ansatz für das Software Engineering bezeichnet wird. Auf die dabei entwickelten, Kompositionsmöglichkeiten wird im Folgenden noch eingegangen.

Die Softwareentwicklungsindustrie hat das Wissen bezüglich der Fähigkeiten objektorientierter Programmiersprachen und die vielfältigen theoretischen Kommunikationsmöglichkeiten zwischen Komponenten sowie die Erfahrungen mit verteilten Systemen genutzt und darauf aufbauend zwei Komponententechnologien entwickelt (siehe Abschnitt 3.2.1) DCOM (COM+) und CORBA mit IIOP für die es bereits produktive betriebswirtschaftliche Anwendungssysteme gibt. Web Services sind dabei, ihren Weg in die Anwendungssystementwicklung zu finden. Die nachfolgenden Ausführungen lassen Web Services unberücksichtigt, am Ende des Abschnittes wird dann eine Reflektion der Erkenntnisse mit der Web Service Technologie durchgeführt.

Die Konzepte, Projekte und Komponenten-Technologien werden im Hinblick auf die oben aufgeführten Kernforderungen untersucht.

- **Explikation der Kunden- und Lieferanten-Schnittstellen**

Es wurde bereits erwähnt, dass auch die Kunden-Schnittstelle eines Anwendungs-elementes ausdrücklich erkennbar sein muss. Die Kunden-Schnittstelle stellt einen so genannten Kommunikationsimport dar und nur dadurch ist eine Vernetzung zwischen Anwendungselementen ersichtlich. In den Konzepten des „Programmieren im Großen“ und der Software-Architektur von Shaw und Garlan [Shaw/Garlan96] sowie den beiden

erwähnten Konfigurationsprojekten ist die explizite Darstellung der Kunden-Schnittstelle vorgesehen. In den industriellen Komponenten-Technologien ist dieses Feature jedoch nicht vorhanden. Es wird zwar im Compilierungsvorgang geprüft, ob ein Methodenaufruf einen eindeutigen Empfänger (Lieferanten) hat[32], allerdings ist die Komponentenvernetzung anhand der Schnittstellenbeschreibung (IDL – Interface Definition Language) nicht ersichtlich. Die für den Konfigurationsvorgang notwendigen Kompositionsstrukturen (Verwendungsnachweise) müssen also separat verwaltet werden, beispielsweise in einem Komponenten-Repository, wie ***TKMS***.

- **Realisierung des Parameterpolymorphismus**

Definition:

> **Polymorphism** aids reuse by allowing newly spezialized components to be used in the same environment as old components without having to modify the calling environment. [Coleman et.al. 94, 216]

Die Bezeichnung „alte“ Komponente muss dabei sinngemäß auf ursprüngliche Komponente übertragen werden, da nicht nur neue, verbesserte sondern auch andere, variierte Komponenten davon betroffen sind.

Es gibt zwei Arten des Polymorphismus [Eisenecker95, Sullo94, Börstler94, 30]:

1. Vererbungs- oder Inklusionspolymorphismus und
2. Parameter- oder signaturgebundener Polymorphismus

Der Parameterpolymorphismus ist für die kompositionale Wiederverwendung einzusetzen. Es geht im Wesentlichen um die Verwendung derselben Schnittstelle für unterschiedliche (polymorphe) Implementierungen. Auf die Möglichkeiten des dynamischen Methodenaufrufs in CORBA [Orfali et.al. 96, 76ff] oder auch auf die Vor- und Nachteile von Füllparametern soll an dieser Stelle nicht eingegangen werden.

Eine Aufgabe muss im Zusammenhang des Parameterpolymorphismus und der Variantenkonstruktion gelöst werden: Der Name des Zielobjektes (Lieferant) ist zur

---

[32] Mit Eindeutigkeit ist hier die korrekte Unterstützung des Methoden-Zuordnungsverfahrens gemeint. Die Möglichkeiten dafür folgen später in diesem Abschnitt.

Entwicklungszeit nicht bekannt, er wird erst zur Konstruktionszeit festgelegt. Daraus ergibt sich auch die nächste Kernforderung.

- **Realisierung eines durch Komposition aktivierten Verbindungs-mechanismus**

Die Komponenten-Technologien DCOM und CORBA haben lediglich für den Verteilungsaspekt einen Transparenzmechanismus für die Variation jedoch nicht. Sowohl in DCOM als auch in CORBA sollte der Name der Ziel-Komponente zur Konstruktionszeit bekannt sein[33]. Es müssen also Lösungen für eine dynamische Zielkomponenten-Zuordnung zur Laufzeit geschaffen werden. Dabei werden die Methodenaufrufe zwischen den Komponenten nicht fest programmiert, sondern in den aufrufenden Funktionen variabel definiert. Erst im Auswahlvorgang der Komposition werden die geeigneten Ziele festgelegt und einem „Dispatcher“ (Versender) für die Laufzeitzuordnung zur Verfügung gestellt. Die Zuordnung der Kunden- zu ihren Lieferanten-Schnittstellen könnte auf der Ebene der Kommunikationstechnologie von einem Composer durchgeführt und auf Korrektheit geprüft werden. Das System von Andersen Consulting [Ning et.al. 94] bietet hierfür eine graphische Lösung an. Fachliche Komponenten werden in einer speziell entwickelten „module interface specification language“ beschrieben. Die Signatur der Schnittstelle wird durch ihre Parameter festgelegt, ihr semantisches Verhalten durch Vor- und Nachbedingungen. Ein Composer kann nun die Schnittstellen mittels Verbindungssymbolen mit unterschiedlichen Eigenschaften verbinden. Als Eigenschaften stehen zur Verfügung [Ning et.al. 94, 89]:

- Synchronisations-Modus (synchron, asynchron, verzögert synchron)
- Kardinalität (ein Empfänger, viele Empfänger, „broadcast“)
- Aufruf-Modus (direkt, indirekt)
- Zuverlässigkeit (Nachrichtenlieferung muss angefordert werden, Nachrichtenlieferung ist garantiert)
- Sicherheit (Authentifizierung, Verschlüsselung)

[33] Object Request Broker bieten einen Service an, anhand des Methodennamens ein Objekt zu suchen, welches diese Methode anbietet. Diese Lösung entspricht jedoch nicht den Anforderungen, dass für einen bestimmten Anwendungsfall (Namensraum) ein ausgesuchtes Anwendungselement für die Erfüllung einer bestimmten Aufgabe ausgewählt wird.

Eine Baugruppenbildung wird dabei durch die Verbindung von Schnittstellen mit einem „Baugruppenrahmen“ dargestellt. Nach erfolgter Verbindungsarbeit werden alle Verbindungen nach folgender Regel geprüft: Ein Kunden-Stecker[34] *C* ist dann kompatibel mit einem Lieferanten-Stecker *S*, wenn die Vorbedingungen von *C* die Vorbedingungen von *S* enthalten und die Nachbedingungen von *S* die Nachbedingungen von *C* enthalten.

Die Parameter müssen vom gleichen Typ sein, wobei Subtypen beim Kunden zugelassen sind.

Ist die Kompatibilitätsprüfung erfolglos, so kann der Composer mittels einer „interconnection specification language“ ein spezielles Verbindungsprotokoll entwickeln um die Interaktion der Komponenten trotzdem zu ermöglichen.

Diese Lösung besitzt einen deutlichen Nachteil: ein Composer ist ein Spezialist im Anwendungsbereich; er hat nicht unbedingt die Fähigkeit, Verbindungseigenschaften zu definieren oder eigenständig Verbindungsprotokolle für inkompatible Schnittstellen zu entwickeln.

Es ist eine ausdrückliche Forderung der ***Methode der semantischen Komposition***, dass nach erfolgter Kompositionshandlung die Komponenten-Kommunikation definiert sein muss, ohne dass ein Composer hierfür noch tätig wird.

Voraussetzung dafür ist, dass zu jeder Kunden-Schnittstelle genau eine Lieferanten-Schnittstelle passt. Diese passende Lieferanten-Schnittstelle kann jedoch von verschiedenen Anwendungselementen implementiert werden (Polymorphismus). Das Ergebnis der Variantenkonstruktion stellt dann je „Methodenaufruf-Gültigkeitsbereich“ (Namensraum) eine eindeutige Zuordnung zwischen Anwendungselementen respektive zwischen Lieferanten und Kunden-Schnittstellen her. Nicht zugelassene Kombinationen müssen durch die Restriktionen eines Anwendungselemente-Beziehungsnetzes (siehe Abschnitt 3.4.4) verhindert werden. Umgekehrt sollten für die Auflösung von Kunden-Konflikten, ermittelt durch vorhandene Beziehungen, Kompositions-Kandidaten vorgeschlagen werden.

---

[34] Im Orginal: „client plug“.

Für eine erfolgreiche Komposition müssen folgende Schnittstellenprüfungen durchgeführt werden:

1. Angeforderte Lieferanten-Schnittstellen müssen eindeutig ermittelt werden können. Die bereits formulierte Anforderung, dass Schnittstellen-Abhängigkeiten geringer ausfallen sollen als inhaltliche Abhängigkeiten zwischen Anwendungselementen, macht diese Prüfung eigentlich überflüssig, denn eine konfliktfreie Berücksichtigung der semantischen Relationen schließt die Lösung der Lieferanten-Konflikte mit ein.
2. Eine Parameter-Typprüfung sollte erfolgen, denn eine Weiterentwicklung der Schnittstellen könnte ihre Signatur verändern.

Zur Laufzeit kann zusätzlich die Erfüllung von Vor- und Nachbedingungen geprüft werden.

Das Prinzip der Implementierung von Kunden-Schnittstellen für genau eine Lieferanten-Schnittstelle und die direkte Kommunikation der Schnittstellen hat auch Nachteile. Es entsteht ein erhöhter Entwicklungsaufwand, denn die Kommunikationssicherheit muss in den jeweiligen Schnittstellenimplementierungen erfolgen. Zudem ist eine permanente Harmonisierung der Schnittstellen erforderlich.

Flexiblere Lösungen bieten hier beispielsweise Komponenten-Interaktions-Protokolle, so genannte „Gluons“ [Pintado95], welche die Kommunikation zwischen den Komponenten zu einem erheblichen Teil außerhalb der Komponenten lösen. Dies führt zu weniger Redundanz in der Entwicklung. Hinzu kommt eine Flexibilisierung der Kommunikationsmöglichkeiten zwischen Komponenten - auch für Kommunikationsbedürfnisse, die zur Entwicklungszeit noch nicht vorhersehbar waren.

Auch Shaw und Garlan [Shaw/Garlan96, 160ff] haben hierfür ein Konstrukt geschaffen: „Connectors“. Mit diesen Connectors lassen sich schon in der Systemarchitektur die Beziehungen zwischen Komponenten auf abstrakter Ebene modellieren.

Die Mächtigkeit dieser eigenständig realisierten „Beziehungskomponenten“ wird durch Justierbarkeit verschiedener Eigenschaften, die sinnvoll für eine flexible Kommunikation zwischen Anwendungselementen sind, zum Ausdruck gebracht. Bisher gibt es

für derartige Beziehungskomponenten in den industriellen Komponenten-Technologien noch keinen Standard.

Eine Abwandlung des hier vorgestellten Konzeptes in Form einer dynamischen Auswahl der einzusetzenden Anwendungselemente zur Laufzeit, abhängig von zuvor festgelegten Kriterien, wird im Anwendungsfall sicher interessant werden. Als Ergebnis des Variantenkonstruktionsprozesses liegen dann alternative Anwendungselemente für eine bestimmte (Teil-)Aufgabe vor sowie die Regeln, bei Erfüllung welcher Kriterien welche Anwendungselemente eingesetzt werden.

Eine Reflektion der oben aufgeführten Erkenntnisse und ihre Anwendung auf die Web Service Technologie führt zu folgendem Ergebnis.

Für eine Explikation der Kunden-Schnittstelle hat auch die Web Service Technologie keine Antwort. Die lockere Kopplung (loose coupling) von Web Services bedeutet ja, dass ein Interface nicht mehr weiß, von welcher anderen Komponente es verwendet wird. Trotzdem ist in einem System mit geschäftskritischer Stabilität und massen-marktfähiger Wartungseffizienz ein Verwendungsnachweis für Schnittstellen unerlässlich. Die Anbieter von servicebasierten Anwendungsplattformen müssen eine geeignete Lösung zur Verwaltung der Web Service Nutzer bereitstellen.

Parameterpolymorphismus ist einer der Stärken von Web Services, denn alle Web Services mit identischer WSDL (Web Service description language)-Beschreibung ihres Interfaces können als Varianten betrachtet werden und sind somit austauschbar. Schnittstellenprüfungen werden bei Web Services automatisch durchgeführt

Leider bietet auch die Web Service-Infrastruktur-Standardisierung keine Lösung für dynamisches Zuordnen der korrekten URI (Uniform Resource Identifier) aufgrund einer Kompositionsentscheidung. Es gibt zwar einen Verzeichnisdienst (UDDI: universal description, discovery and integration) [Booth et.al. 04], der alle Web Services mit ihrer WSDL Beschreibung und ihren URI’s für eine bestimmte Domäne vorrätig hält, allerdings ist UDDI mehr auf kollaborative Geschäftsprozesse fokussiert mit Interaktiver Auswahl von geeigneten Web Services bzw. einfachen Abfragen von Web Service-Eigenschaften. UDDI hat nicht den Anspruch Web Service-Kommunikation basierend auf einem Anwendungsbauplan zu organisieren.

3.4 Aufbau eines Ordnungssystems für Anwendungselemente

Die Beschreibung der ***Methode der semantischen Komposition*** besteht bisher aus der Definition ihrer Konstruktionselemente, der Erläuterung spezieller Konstruktionsprinzipien und der Darstellung des Aufbaus eines Baukastensystems für Anwendungselemente. Für ein ingenieurgerechtes Konstruieren fehlt jetzt nur noch eine Ordnungssystematik, wie sie sich in technischen Konstruktionsbereichen etabliert hat. So ein Ordnungssystem hat sowohl einen beschreibenden als auch einen gliedernden Charakter. In den nachfolgenden Abschnitten wird ein Transfer der Erfahrungen technischer Konstruktionsarbeit mit Ordnungssystemen auf die Anwendungssystemkonstruktion vollzogen. Der Erfolg technischer Ordnungssysteme ist unter anderem auf die Normierung der Fachsprache in den jeweiligen Anwendungsbereichen zurückzuführen. Dieser so wesentliche Aspekt für die Entwicklung eines Ordnungssystems für Anwendungselemente wird erst zum Ende dieses Kapitels erklärt, da die Instrumente erst klar definiert sein sollten, bevor ihr terminologischer Aufbau vermittelt wird.

3.4.1 Technologieunabhängige Beschreibung von Anwendungselementen

Eine Beschreibung von Anwendungssystemkomponenten kann unterschiedlichsten Intentionen folgen. Einerseits steht das Retrieval von Komponenten in einer Bauteil-Bibliothek oder einem Repository im Vordergrund und andererseits die Dokumentation von Softwareentwicklungs- und Contententwicklungsergebnissen und ihren Beziehungen untereinander. Überwiegend sind die Beschreibungskonzepte geprägt durch die Implementierungstechnologie der Komponenten oder die Konzeption des Verwaltungswerkzeuges [Prieto-Diáz/Freeman87, Ostertag et.al. 92, Heß/Scheer92, Chen93, Hobbs94, Convent/Wernecke94, Poulin/Werkman96, Henninger97]. Bei Turowski wird eine ganzheitliche Spezifikation von so genannten Fachkomponenten beschrieben. Das sind Komponenten, die Dienste einer betrieblichen Anwendungsdomäne anbieten [Turowski01, Turowski02]. Die von Turowski und anderen Mitgliedern des Arbeitskreises vertretenen Beschreibungsebenen sind sehr gut geeignet für eine marktorientierte Sicht auf Fachkomponenten [Turowski02].

In der vorliegenden Arbeit wird eine technologieunabhängige Beschreibung von Anwendungselementen vorgestellt. Sie hat zum Ziel, eine Variantenkonstruktion von Standard-Anwendungssystemen für Branchen- und Kundenlösungen bestmöglich zu

fördern. Eine Ableitung der Schnittstellendefinitionen aus den inhaltlichen Beschreibungen der Komponenten wird hier nicht weiter betrachtet.

Die nachfolgend aufgeführten Beschreibungsformen basieren zum Teil auf den Erfahrungen, die mit dem Komponenten-Management-System ***TKMS*** gemacht wurden (siehe Kapitel 4). Außerdem sollen sie speziell die Komponenteneigenschaften Kapselung, Austauschbarkeit und Komposition unterstützen.

Beschreibungsformen für Anwendungselemente:

- Interne Struktur: Aufbaustruktur von Anwendungselementen aus Creatorelementen
- Externe Struktur: Baugruppen- und Kontextbeziehungen
- Dokumentation: Anforderungen, Funktionsbeschreibung, Präsentationsbeschreibung, Anwenderdokumentation, Entwurf-, Design- und Implementierungsergebnisse
- Verwaltungsinformationen: Bibliotheksinformationen und betriebswirtschaftliche Informationen für Komponenten
- Aufgabenorientierte Eigenschaften in Form von Beschreibungsvektoren aus Merkmalausprägungen

Eine technologieunabhängige Beschreibung von Anwendungselementen setzt voraus, dass das Wesen der Bauteile, dem von Komponenten entspricht. Insofern wird die Technologieunabhängigkeit durch die ausdrückliche Darstellung von Kompositions- und Kommunikationsbeziehungen als „Externe Struktur“ nicht verletzt. Die „Interne Struktur“ ist nur soweit unabhängig, als sie ebenfalls ausschließlich Kompositionsbeziehungen darstellt. Werden hierbei Vererbungsbeziehungen, die Nutzung gemeinsamer Datenspeicher, funktionale Strukturen, alles was sich nicht auf eine Kommunikation über Schnittstellen zurückführen und durch Partition zerlegen lässt, dargestellt, so ist die Grenze hin zur Technologieabhängigkeit überschritten.

Dabei ist diese Grenze nicht der bestimmende Faktor für die Bezeichnung „Komponente“. Ein Datenelement, eine globale Funktion, ein Objekt einer objektorientierten Programmiersprache oder ein Design Pattern kann durchaus als

(technologieabhängige) Komponente im Sinne von Bestandteil verstanden und sinnvoll in einem Komponentenverwaltungssystem (Repository) gespeichert werden.

Ein Anwendungselement ist mit allen Ergebnissen des Softwareentwicklungsprozesses - sofern vorhanden - als Ganzheit zu sehen. Deshalb ist die „Dokumentation“ als Struktur (beispielsweise in Form von Dokumentationsbeziehungen) technologie-unabhängig; die Dokumentationsergebnisse bzw. –inhalte sind es jedoch nicht mehr.

Eine gemeinsame sprachliche Basis für die Kommunikation zwischen Anwender und Entwickler von Softwareprodukten wird durch die gemeinsame Nutzung von „Begriffen“ begründet.

Definition:

> **Begriff**, ein durch Abstraktion gewonnener abstrakter Gegenstand, durch den andere Gegenstände oder Sachverhalte aufgrund bestimmter Eigenschaften und/oder Beziehungen klassifiziert werden. Begriffe werden durch Termini dargestellt. Sie lassen sich wie Mengen definieren: (a) extensional durch Aufzählen der Objekte, die unter einen bestimmten Begriff fallen, und (b) intensional durch Angabe ihrer spezifischen Merkmale. Merkmale sind wieder Begriffe [in Anlehnung an Bußmann90].

Werden also Anwendungselemente als Begriffe einer Konstruktionssprache für Anwendungssoftware verstanden, so können sie intensional über Merkmale beschrieben werden. Wobei die Bezeichnung Merkmal nicht völlig korrekt ist. Denn die so bezeichneten Merkmale sind eigentlich Ausprägungen. Diese Ausprägungen werden implizit nach Merkmalen klassifiziert. Als Beispiel ist hier ein Exemplar der Gegenstandsklasse „Drucker“ in einem Verkaufskatalog beschrieben:

Tintenstrahldrucker, 720 x 600 dpi, 10 Blatt/min., s/w

In dieser Beschreibung werden vier Ausprägungen (z.T. inkl. Einheit) aufgeführt, deren Merkmale der Experte einfach zuordnen kann.

Merkmal 1: Drucktechnik	→	Ausprägung 1: Tintenstrahl-Technik
Merkmal 2: Druckauflösung	→	Ausprägung 2: 720 x 600 dots per inch

Merkmal 3: Druckgeschwindigkeit → Ausprägung 3: 10 Blatt pro Minute
Merkmal 4: Farbdruckeigenschaft → Ausprägung 4: schwarz-weiß

Damit Beschreibungen von Anwendungselementen mittels Merkmalen nicht nur von Experten des jeweiligen Fachgebietes korrekt verwendet werden können, wird die Bezeichnung „Merkmalausprägung“ als Tupel für die Beschreibung einer Anwendungselemente-Eigenschaft festgelegt (z.B. Farbe/rot) [Meinl90].

Die Beschreibung aller Eigenschaften eines Anwendungselementes kann als Vektor bezeichnet werden [Ashby74, 55ff]. Seine Komponenten (Dimensionen) sind die Merkmale und seine Werte entsprechen den Ausprägungen. Jede Komponente des Vektors kann wiederum ein Vektor sein, so sind auch mehrwertige Ausprägungen eines Merkmals als zulässig definiert.

Die Möglichkeit, Gegenstände, Begriffe oder Softwareartefakte durch aufgelistete Begriffspaare - bestehend aus Eigenschaftsbezeichnung und Eigenschaftswert - zu beschreiben, hat sich schon in vielen wissenschaftlichen Bereichen etabliert und als sinnvoll erwiesen. Beispielsweise verwendet die Semantik in der Sprachwissenschaft intensionale Begriffsbestimmungen [Bußmann90], und die Bedeutungslehre der Psychologie setzt Osgoods „semantisches Differential“ ein [Osgood et.al. 78]. Datenelemente werden durch ein deskriptorbasiertes Beschreibungssystem verwaltet [Brenner88]. Auch in der semantischen Datenmodellierung spiegelt sich dieses Prinzip wider. Attribute und ihre Werte (Ausprägungen) werden für die Beschreibung von Informationsobjekten (Objekttypen) eingesetzt [Ortner/Söllner89]. Für wiederverwendbare Ergebnisse des Softwareentwicklungsprozesses hat sich die Facettenklassifikation etabliert [Prieto-Diáz/Freeman87]. Im Fachgebiet des Software-Konfigurations-Managements hat sich die Beschreibung mittels „feature-terms“ als besonders geeignet erwiesen [Zeller97, 63ff, Zeller/Snelting97, 5ff]. In der Wissensrepräsentation haben sich Frames für dieses Prinzip der Begriffsbeschreibung etabliert [Reimer91, 159ff].

Der Erfolg dieser Form der Begriffsbeschreibung liegt darin begründet, dass das semantische Gedächtnis des Menschen Begriffe ebenfalls über Merkmalmodelle verwaltet [Mayer79, Eckes91]. Semantische Netze sind den Merkmalmodellen im Wesentlichen äquivalent.

Ein wichtiger Aspekt dieser Beschreibungsvektoren ist, dass Vektoren mit identischen Merkmalen zu Klassen zusammengefasst werden können. Die verschiedenen Merkmalausprägungen einer Klasse bilden somit die Beschreibungen für verschiedene Objekte dieser Klasse. Die oben aufgeführten Beschreibungssysteme fassen Merkmalgruppen ebenfalls zu Klassen zusammen, ob sie diese Klassen nun ausdrücklich benennen oder nicht. Frames sind beispielsweise direkte Klassen-bezeichnungen ebenso wie die Objekttypen in der Datenmodellierung. In Prieto-Diáz' und Freemans Facettenklassifikation stellen die Klassen ganze „application domains“ dar, und im Bereich des Konfigurations-Managements werden Klassen dynamisch über gemeinsame „features“ gebildet.

Für die Klassifizierung von Anwendungselementen wird das Prinzip der **Sachmerkmalleisten** vorgeschlagen und im nachfolgenden Abschnitt weiter ausgeführt.

Mit den Merkmalausprägungs-Vektoren steht eine Struktur zur Beschreibung aller Eigenschaften von Anwendungselementen zur Verfügung. Der Inhalt der Merkmale und ihrer Ausprägungen muss für eine technologieunabhängige Beschreibung auf fachliche Eigenschaften beschränkt bleiben. Die expliziten Eigenschaften der Anwendungselemente, welche unmittelbar den Auswahlvorgang des Composers unterstützen, sind in jedem Fall fachlicher Art und somit technologieunabhängig. Das ist insofern tautologisch, als sie genau für diesen Zweck ausgewählt wurden, damit der Composer von Technikentscheidungen befreit wird. Merkmale, welche implizite Eigenschaften beschreiben, können sowohl technologieunabhängig als auch technologiespezifisch sein. Das ist abhängig von der Art der Filtersysteme bzw. welche Eigenschaften sie für ihren Filtervorgang verwenden (siehe Abschnitt 3.3.7).

3.4.2 Sachmerkmalleisten für Anwendungselemente

Für die ***Methode der semantischen Komposition*** wurden die in technischen Ingenieurbereichen etablierten Sachmerkmalleisten (siehe Abschnitt 2.1.5), zur Beschreibung und Klassifizierung von Anwendungselementen, als geeignet ausgewählt. Mit ihren Merkmalen und Ausprägungen können die Eigenschaften der Anwendungselemente gut repräsentiert werden.

Folgende Grafik soll die Beschreibung eines Anwendungselementes verdeutlichen.

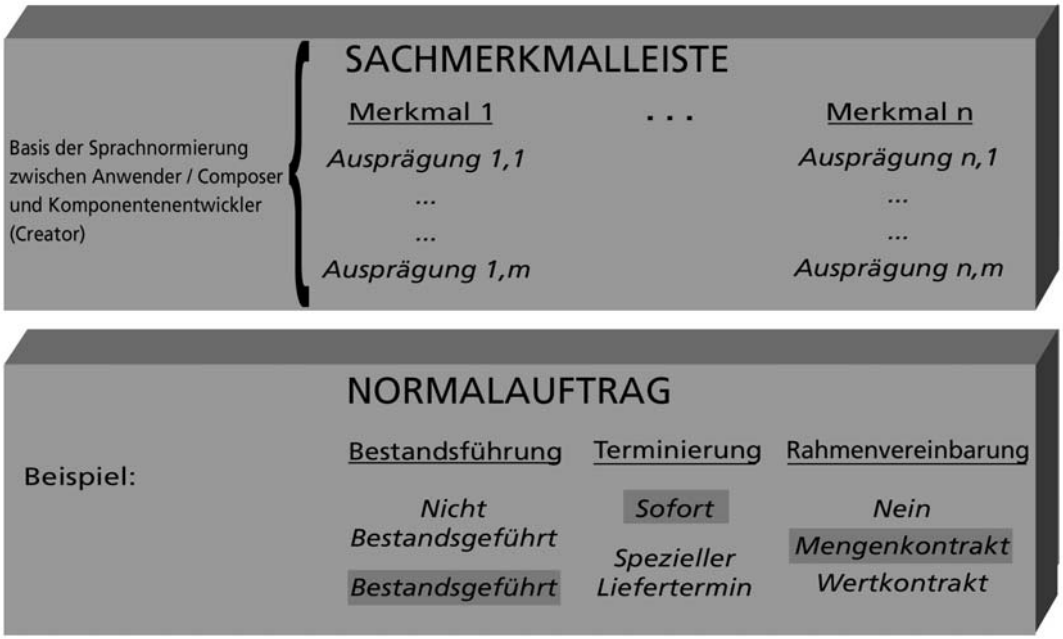


Abbildung 26: Sachmerkmalleiste für Anwendungselemente

Sachmerkmalleisten sind die Repräsentanten der Anwendungselemente im Ordnungssystem. Ein Merkmal (bzw. eine Kombination von Merkmalen) repräsentiert dabei eine Gruppe von Eigenschaften (Merkmalausprägungen). Eine Sachmerkmalleiste repräsentiert als Struktur eine Gruppe von Merkmalen und als Begriff eine Gruppe von Anwendungselementen, die sich mit den Merkmalausprägungen dieser Sachmerkmalleiste beschreiben lassen.

Sachmerkmalleisten bilden die Kommunikationsbasis sowohl für Anwender und Composer als auch für Entwickler. Hinsichtlich der Klassifizierung von Anwendungselementen haben die differenten Merkmalausprägungen einer Sachmerkmalleiste folgende Bedeutung:

> Analog zu Varianten in der technischen Konstruktionslehre (Abschnitte 2.1.4 und 2.1.5) werden **Anwendungselemente-Varianten** durch unterschiedliche Ausprägungen derselben Merkmale einer Sachmerkmalleiste (Klasse) unterschieden.

Die Einordnung der Anwendungselemente in die Merkmalausprägungen von Sachmerkmalleisten stellt ihre Eigenschaftsbeschreibungen in Form von Eignungsprofilen dar. Umgekehrt ist bei der Lösungsauswahl während des

Konstruktionsprozesses für Anwendungen aus Komponenten die Festlegung von Merkmalausprägungen für die gewünschte Lösung ein Anforderungsprofil an Anwendungselemente. Ist im Sortiment der Anwendungselemente keine entsprechende Lösung vorhanden, kann mit diesem Anforderungsprofil eine sehr konkrete Herstellungsanforderung an die Creator weitergegeben werden. Diese Anforderung enthält implizit die Lösungsnähe zu anderen Anwendungselementen dieser Sachmerkmalleiste und bietet somit weitere Vorgaben für ihre Herstellung.

Für den Auswahlvorgang im Konstruktionsprozess werden in einer Sachmerkmalleiste nur diejenigen Merkmale dargestellt, die so genannte explizite Eigenschaften repräsentieren. Weitere implizite Eigenschaften dieser Klasse von Anwendungselementen sind ebenfalls der Sachmerkmalleiste zugeordnet, werden jedoch nur auf Anforderung angezeigt.

Die Anzahl der expliziten Merkmale wächst mit der Vielfalt im Sortiment, denn alle Anwendungselemente müssen durch ihre Merkmalausprägungen unterscheidbar sein. Das gilt allerdings nur für Anwendungselemente, die verschiedene Lösungen anbieten. Anwendungselemente mit gleichen Anwendungslösungen auf der fachlichen Ebene haben dieselben Merkmalausprägungen. Ihre Unterscheidung muss dann über implizite technologieabhängige Eigenschaften erfolgen.

In der Regel werden die Merkmalausprägungen einer Sachmerkmalleiste umfangreicher sein als ihr Sortiment, da bestimmte Kombinationen nicht als Anforderung erwartet werden. Hierbei zeigt sich die Anwendernähe dieses Systems, denn falls sich ein Anwender dennoch bestimmte Merkmalausprägungen wünscht, kann er das Anforderungsprofil „seiner“ Variante direkt über die Merkmalausprägungen definieren.

Sachmerkmalleisten sind nach DIN [DIN 4000-92] keine vollständigen Spezifikationen. Folglich sind nicht alle Eigenschaften von Gegenständen in einer Sachmerkmalleiste enthalten. Die Beschreibungstiefe sollte so groß sein, dass sie für den vorgesehenen Verwendungszweck ausreicht. Hier liegt das Problem der Sachmerkmalleisten-Modellierung. Die Festlegung, welche Merkmale noch Bestandteil einer Sachmerkmalleiste sein sollen, ist von vielen „weichen“ Faktoren abhängig (z. B. Erfahrung, Benutzerwissen und weitere). Eine Kategorisierung von Merkmalen in Gliederungs-, Beschreibungs- und auswahlerleichternde Merkmale wird im nächsten Abschnitt,

nachdem die Katalogstruktur erläutert wurde, weiter ausgeführt und kann dabei helfen, die richtigen Modellierungsentscheidungen zu treffen.

Das Anwendungselemente-Beispiel aus Abschnitt 3.3.3 unterstützt im Folgenden die Erläuterung weiterer Aspekte von Sachmerkmalleisten. Dazu wird in folgender Tabelle noch einmal das Gegenstandsmuster der *Kundenbestellung für Lagerartikel* gezeigt.

Merkmale		**Ausprägungen**
Bestandsführung	–	Bestandsgeführt
Produktbereitstellung	–	Kommissionierung
Rahmenvereinbarung	–	Auftragsrabatt
Terminierung	–	Spezieller Liefertermin
Arbeitsteilung	–	Zentralisierte Bearbeitung
Kundenart	–	Firmenkunden

Tabelle 6: Gegenstandsmuster der *Kundenbestellung für Lagerartikel*

Dieses Gegenstandsmuster stellt also den Beschreibungsvektor für ein Anwendungselement dar, wobei die Dimensionen des Vektors den Merkmalen (Bestandsführung, ...) entsprechen und die Ausprägungen den Werten je Dimension (Bestandsgeführt, ...). Die möglichen (zugelassenen) Wertekombinationen, die ein Vektor annehmen kann, werden als seine „Freiheitsgrade“ bezeichnet [Ashby74].

Die **Variabilität eines Baukastensystem** wird durch das Produkt der Freiheitsgrade all seiner Beschreibungsvektoren[35], abzüglich der nicht zugelassenen Baustein-Kombinationen, bestimmt.

Das Beispiel wird um zwei Merkmale erweitert, die eine Vertiefung des Verständnisses von Sachmerkmalleisten fördern: *Produktlieferung* und *Artikelart*.

Die Sachmerkmalleiste **Kundenbestellung** wird nun in ihrem vollen Umfang dargestellt:

[35] Nur von elementaren Bausteinen.

Abbildung 27: Sachmerkmalleiste *Kundenbestellung*

Anhand dieser Sachmerkmalleiste können verschiedene Aspekte der Subsumtion von Anwendungselementen unter Sachmerkmalleisten gezeigt werden. Zuvor wird jedoch das Gegenstandsmuster (bzw. Eignungsprofil) der *Kundenbestellung für Lagerartikel* mit den erweiterten Merkmalen visualisiert.

Abbildung 28: Gegenstandsmuster der *Kundenbestellung für Lagerartikel*

Dieses Beispiel hilft, folgende Aspekte zu verdeutlichen:

1. zulässige und unzulässige Komplexmerkmale
2. notwendige Abhängigkeiten zwischen Merkmalausprägungen innerhalb einer Sachmerkmalleiste
3. Multigegenstände
4. explizite und implizite Eigenschaften
5. Bedeutung von Nullzuständen

Ad 1) zulässige und unzulässige Komplexmerkmale:

Ein Komplexmerkmal ist laut Meinl „eine **additive** Zusammenfassung mehrerer ‚Einzelmerkmale' zu einem Merkmalbegriff"[36] [Meinl90, A9]. In dem o.a. Beispiel ist *Produktlieferung* ein Komplexmerkmal, welches durch die additive Zusammenfassung von *Bestandsführung* und *Produktbereitstellung* erzeugt werden kann. Folgende Tabelle stellt die Erzeugungsmöglichkeiten dar:

Produktlieferung		**Bestandsführung**		**Produktbereitstellung**
Lager	⇔	Bestandsgeführt	∨	Kommissionierung
Produktion	⇔	Bestandsgeführt	∨	Produktionsauftrag
Auf Strecke	⇔	Nicht Bestandsgeführt	∨	Keine

Tabelle 7: Beispiele für Komplexmerkmale

Komplexmerkmale sind gemeinsam mit ihren Einzelmerkmalen in derselben Sachmerkmalleiste **nicht zugelassen!**

Komplexmerkmale in elementaren Anwendungselementen sollten vermieden werden, denn sie schränken die Flexibilität einer möglichen Variantenbildung ein. Das Merkmal *Produktlieferung* muss folglich aus der Sachmerkmalleiste **Kundenbestellung** entfernt werden.

[36] Einzelmerkmale können natürlich in aufsteigender Abstraktion selbst auch Komplexmerkmale sein; dies hat jedoch keinen Einfluss auf die hier getroffenen Aussagen.

Komplexmerkmale sind geeignet für Sachmerkmalleisten zur Beschreibung von Anwendungselemente-Baugruppen. Baugruppen befinden sich auf anderen Abstraktionsebenen als ihre Elemente. Der reduzierte Auflösungsgrad kann nicht nur die Partition der Baugruppen verbergen, sondern durch die Verwendung von Komplexmerkmalen auch die Einzelmerkmale der Elemente. Empfehlungen der ***Methode der semantischen Komposition*** für die Eigenschaftsbeschreibung von Baugruppen werden im Abschnitt 3.4.3 formuliert.

Ad 2) notwendige Abhängigkeiten zwischen Merkmalausprägungen innerhalb einer Sachmerkmalleiste:

Zwischen der *Bestandsführung* und der *Produktbereitstellung* besteht eine wechselseitige Abhängigkeit, denn nur die o. a. Ausprägungskombinationen sind zulässig. Dies ist scheinbar ein Widerspruch zu der Forderung, „unabhängige Variabilitätsfaktoren“ für elementare Anwendungselemente zu finden.

Es gibt nun zwei Möglichkeiten:

1. Kundenbestellungen sind keine elementaren Anwendungselemente. Im Zubehörverkauf-Beispiel wird jedoch die Kundenbestellung als elementares Anwendungselement vorausgesetzt.
2. Eine Suche nach vollständig unabhängigen Faktoren ist nicht immer erfolgreich. Dörner hat diesen Umstand angesprochen [Dörner87, 111] und als „Bündeltheorie“ deklariert[37]. Dabei wird vermutet, dass sich Faktoren als Bündel von Elementarfaktoren darstellen.

Abhängigkeiten zwischen Merkmalausprägungen innerhalb einer Sachmerkmalleiste stellen kein Problem dar. Sie werden genauso wie andere inhaltliche Abhängigkeiten in dem Beziehungsnetz der semantischen Relationen verwaltet. Es muss nur sorgfältig geprüft werden, ob bei elementaren Anwendungselementen die Merkmalbestimmung bestmöglich durchgeführt wurde. Kriterien sind dabei: Eindeutigkeit, Klarheit und Flexibilität bezüglich möglicher Erweiterungen.

---

[37] Dörner hat als Vertreter der Bündeltheorie Thomsons „Stichprobentheorie“ genannt und aus [Hofstätter66, 175f] zitiert.

Ad 3) Multigegenstände:

Ein Multigegenstand ist dadurch gekennzeichnet, dass er bei mindestens einem Merkmal mehrere Ausprägungen (Eigenschaften) besitzt. Das Beispiel-Anwendungselement ist durch den Besitz von drei Ausprägungen des Merkmals *Artikelart* ein Multigegenstand.

Die Frage, ob ein Gegenstand ein Multigegenstand ist, ist nicht trivial zu beantworten. Zunächst einmal werden nur elementare Anwendungselemente betrachtet. Eine Sachmerkmalleiste für elementare Anwendungselemente muss daraufhin untersucht werden, ob sie Komplexmerkmale enthält. Ist dies der Fall, so ist es wahrscheinlich, dass Anwendungselemente mit Merkmalausprägungen von Komplexmerkmalen „versteckte“ Multigegenstände sind. Sind keine Komplexmerkmale in einer Sachmerkmalleiste vorhanden, können Multigegenstände durch die Mehrfachzuordnung von Ausprägungen eines Merkmals identifiziert werden. Allerdings gibt es auch hierbei Ausnahmen. Besitzen alle Anwendungselemente dieser Sachmerkmalleiste immer dieselbe Ausprägungskombination, die für die Identifizierung als Multigegenstand verantwortlich ist, so kann dies ein Indiz dafür sein, dass hier die Ausprägungen unsachgemäß festgelegt wurden. Unter Umständen wurde eine elementare Eigenschaft in verschiedene Ausprägungen aufgeteilt. Der Fall muss sorgfältig geprüft werden, denn werden die multiplen Ausprägungen durch eine einzige ersetzt, obwohl lediglich das Angebot des bisherigen Sortimentes einen falschen Eindruck vermittelte, so ist wiederum die Flexibilität der Variantenbildung ungerechtfertigt eingeschränkt.

Werden nun Baugruppen betrachtet, so ist die Schaffung von Multigegenständen durch Baugruppenbildung ein häufig vorkommendes Ereignis. Eine Identifizierung als Multigegenstand ist abhängig von der Wahl der Merkmale. Für Komplexmerkmale gilt, dass ein Multigegenstand nach obiger Regel nicht mehr identifiziert werden kann. Werden Einzelmerkmale verwendet, so ist eine Variantenbildung innerhalb der Sachmerkmalleiste flexibler gestaltet; es können jedoch Baugruppen mit Multigegenstandscharakter entstehen.

Die Bestimmung ihrer Eigenschaften ist wesentlich für die Gestaltung von Anwendungselementen, sowohl auf fachkonzeptioneller Ebene als auch auf der Herstellungsebene. Falsche Eigenschaftsfestlegungen können das Komponenten-Design für die Herstellung negativ beeinflussen. Außerdem kann die Komposition der

Anwendungselemente zu sinnvollen Baugruppen erschwert werden. Von der passenden Eigenschaftsbestimmung für elementare Anwendungselemente hängt es unter anderem ab, ob das Optimum an Gesamtsystemflexibilität erreicht werden kann.

Der Einsatz von Multigegenständen sollte für elementare Anwendungselemente - soweit möglich - vermieden werden. Auch für Baugruppen niederer Abstraktionsebenen ist diese Empfehlung gültig. Ist die Vereinigung multipler Eigenschaften in einer Baugruppe ein ausdrücklicher Anwenderwunsch und können die Eigenschaften durch die Komposition elementarer Anwendungselemente realisiert werden, so ist das Konzept der dynamischen Konfiguration zur Laufzeit bevorzugt einzusetzen, da es mehr Flexibilität als Multigegenstände verspricht (siehe auch Abschnitt 3.3.8.2).

Multigegenstände sind nicht grundsätzlich negativ zu betrachten. Die Folgen der Multigegenstandsbildung müssen jedoch expliziert werden. Lässt es sich beispielsweise nicht vermeiden, die *Artikelart*-Eigenschaften in Multigegenständen zu vereinen, so muss bei der Verarbeitung der Kundenbestellung die Information, welche Artikelartverarbeitung stattfinden soll, dem System zur Verfügung stehen.[38]

Für die Erzeugung dieser Information ist dann beispielsweise der Artikel „zuständig“, d.h. die Artikelinstanz selbst muss Eigenschaften besitzen (z.B. im Artikelstamm hinterlegt), die Rückschlüsse auf die Verarbeitung zulassen. Für diese Form der Informationsbereitstellung müssen Artikelverarbeitungsinformationen erfasst und laufend aktualisiert werden. Folgende Fragestellungen müssen geklärt werden:

- Wer ist für die Informationen verantwortlich?
- Ist die Informationszuordnung eindeutig?

Unter Umständen ist ein Artikel fallweise der einen oder einer anderen Artikelart zuzuordnen.

- Wie kann dafür eine Geschäftsregel implementiert werden?

[38] Auch hier gilt, dass die nachfolgenden Aussagen als Beispiel für die Darstellung der Problematik dienen. Für eine „reale“ Anwendungsentwicklung müssten die Inhalte viel detaillierter diskutiert werden.

Es entstehen viele dieser Verarbeitungsinformationen, speziell bei Stammdaten werden sie schnell sehr umfangreich und sind dann aufwändig zu erfassen und zu pflegen.

Eine andere Lösung stellt die Informationserzeugung durch den Anwender dar. Jede Interaktion mit dem Anwendungssystem kann Informationen dieser Art liefern. Beispielsweise kann dies eine Auswahlaktion im Menüsystem sein oder die ausdrückliche Beantwortung einer Frage (Beispiel):

- Ist der zu verarbeitende Artikel ein Set-Artikel oder ein Standard-Artikel?

Zudem können aktive Anwenderentscheidungen bezüglich der Aufgabenerfüllung auch in die Arbeitsablaufsteuerung (Workflow-/Prozess-Management-System) integriert werden.

Die Form der Fragestellung und Beantwortung ist von der Gestaltung des User Interface abhängig. Überhaupt kann durch ein gutes User Interface Design mit geeigneten Entscheidungsstellen einem Einsatz von Multigegenständen entgegengewirkt werden, denn Anwenderwünsche bezüglich multiplen Anwendungselemente-Eigenschaften lassen sich häufig auf den Wunsch nach Bedienungskomfort zurückführen. Anwender wünschen sich, dass alle Systemfunktionen leicht erreichbar und produktivitätssteigernd zusammengestellt sind und nicht durch Komponentisierung auf verschiedene Arbeitsbereiche verteilt werden.

Ad 4) explizite und implizite Eigenschaften:

In Abschnitt 3.3.7 wurden die expliziten und impliziten Eigenschaften schon einmal angesprochen. Die ***Methode der semantischen Komposition*** hat das Primärziel, den Kompositionsvorgang für den Composer so optimal wie nur möglich zu gestalten. Dazu gehört es auch, die Informationen, die zu einer Kompositionsentscheidung notwendig sind, auf ein Mindestmaß zu reduzieren und bei Bedarf auf einen sinnvollen Umfang schrittweise zu erweitern.

Die folgende Grafik zeigt eine Dreiteilung der Beschreibungsinformationen aus der Sicht des Kompositionsprozesses. Daran soll die geeignete schrittweise Informationsversorgung des Composers gezeigt werden.

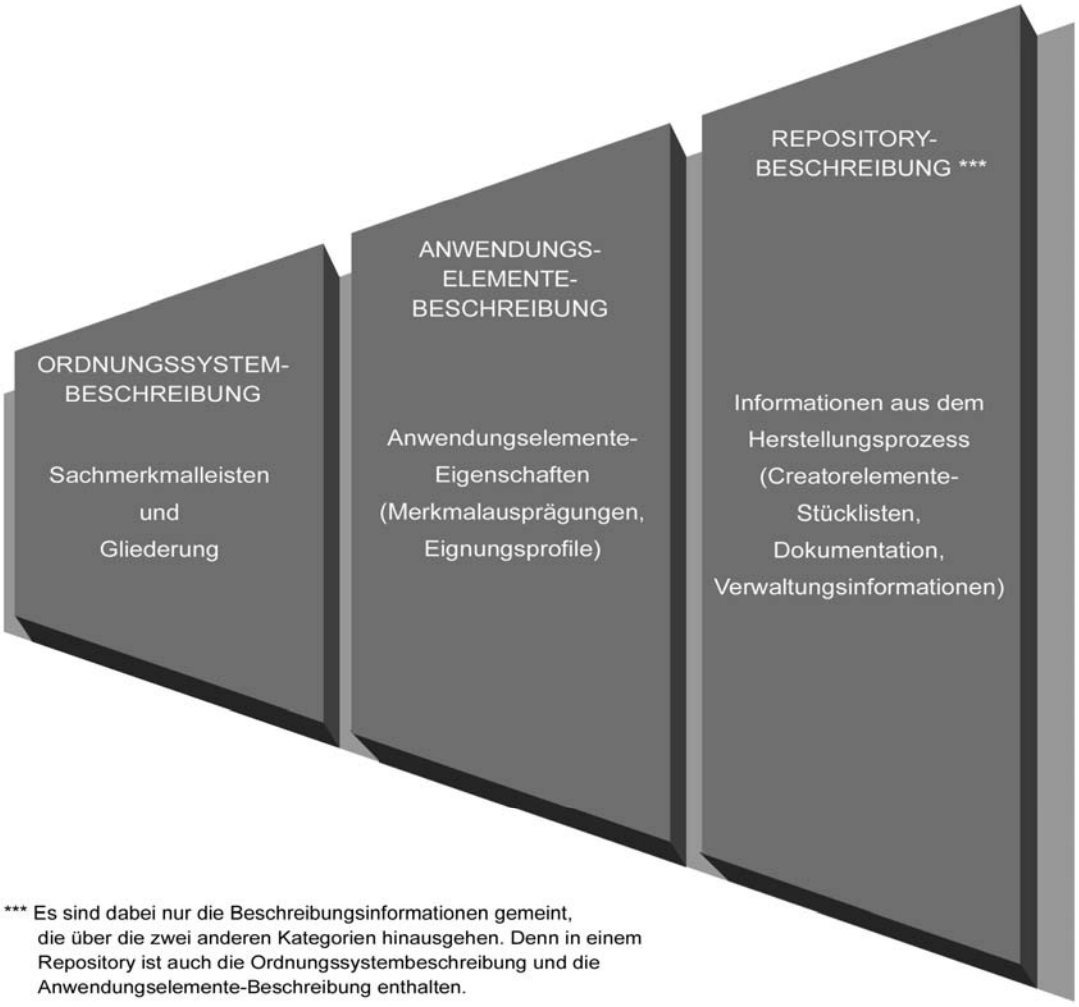


Abbildung 29: Beschreibungsinformationen für den Kompositionsprozess

Die Ordnungssystemstrukturen liefern dem Composer die nötigen Baugruppen- und Kontext-Informationen. Damit kann sich der Composer Wissen über die Einordnung des betrachteten Anwendungselementes in ein Anwendungssystem aneignen. In einem weiteren Schritt schaut er sich das Eignungsprofil des speziellen Anwendungselementes an. Sind die bisher erhaltenen Informationen für eine positive oder negative Kompositionshandlung nicht ausreichend, so kann der Composer im Repository Detailinformationen zu diesem Anwendungselement erhalten. Er kann sich eine umfangreiche Funktionsbeschreibung des Anwendungselementes ansehen. Er kann außerdem die Bedienungsdokumentation des Anwendungselementes zu Rate ziehen. Weiter kann er mögliche User Interface-Lösungen betrachten. Dies geht so weit, dass der Composer auch Informationen über Design und Implementierung des Anwendungselementes einsehen kann, falls er die entsprechenden Berechtigungen besitzt.

Dieses Vorgehen stellt eine gute Abstufung der Informationsfülle dar. Allerdings bieten Sachmerkmalleisten - und folglich auch die Eignungsprofile ihrer Anwendungselemente - Merkmale an, die für eine umfangreiche Eigenschaftsbeschreibung der Anwendungselemente sinnvoll, für Auswahlentscheidungen im Konstruktionsprozess jedoch nicht erforderlich sind.

Auswahl heißt, es wird zwischen mehreren Alternativen eine Entscheidung getroffen. Dazu sind nur die unterscheidenden Merkmale und Ausprägungen notwendig. Alle anderen Informationen führen zu einem Informationsüberfluss und sollten für den ersten Schritt der Informationsbeschaffung verborgen bleiben. Die immer offen liegenden Eigenschaften werden „explizite“ Eigenschaften genannt. Die verborgenen Eigenschaften werden „implizite“ Eigenschaften genannt. Diese Unterteilung gilt für Sachmerkmalleisten und für Eignungsprofile.

Die Arbeitsweise eines Composers sieht also vor, dass zuerst nur die expliziten Eigenschaften betrachtet werden und erst für weiterführende Informationsbedürfnisse auch die impliziten Eigenschaften zugänglich sein sollen.

Implizite Eigenschaften wurden als „für die Auswahlentscheidung nicht erforderlich“ bezeichnet. Das trifft nicht nur für Eigenschaften zu, die aufgrund eines reduzierten Sortimentes einheitlich für jedes Anwendungselement sind und dadurch die Auswahl nicht beeinflussen. Es gibt Eigenschaften, die für eine ausführlichere Beschreibung als Grundlage für die Spezifikation erfasst werden, aber nicht mehr zur Beschreibungstiefe des Composers gezählt werden. Hinzu kommen noch Eigenschaften, die eine so genannte Baureihendifferenzierung darstellen. Aus der Sicht des Composers stellt sich ein Anwendungselement als eine Einheit dar. In Wirklichkeit kann diese „Einheit“ mehrere verschiedene Implementierungsvorschriften besitzen. Typische Vertreter von Baureihen sind die Implementierungen für unterschiedliche Betriebssysteme. Ein Composer sollte während der aufgabenorientierten Komposition von solchen technischen Entscheidungen befreit sein.

Für die Filterung von Anwendungselementen aus dem Angebot des Baukastensystems werden ebenfalls implizite Eigenschaften eingesetzt. Dadurch ist es für Filtersysteme möglich, neben semantischen Relationen auch direkt auf implizite Eigenschaften für die Zulassung und Selektion von Anwendungselementen zuzugreifen. Viele implizite Eigenschaften sind für eine Baureihendifferenzierung zuständig und werden gleichfalls für Filtervorgänge verwendet. Besitzt ein Anwendungselement eine bestimmte implizite

Eigenschaft nicht, so wird es ausgefiltert. Ist die Eigenschaft vorhanden, so wird nach dem Baureihenprinzip vom System die richtige Implementierung vorgenommen. Der Composer muss hier keine Entscheidung treffen.

In dem o.a. Beispiel könnte die *Arbeitsteilung* als implizite Eigenschaft festgelegt werden. In dem Filtersystem für Organisationsmodellierung wird festgelegt, dass im Unternehmensbereich „Einkauf“ zentral alle Beschaffungsvorgänge bearbeitet werden. Für die Auswahl der richtigen Anwendungslösung für eine Kundenbestellung muss bzw. kann der Composer die Entscheidung „zentralisierte versus dezentralisierte Bearbeitung“ nicht mehr treffen. Anwendungselemente, die für eine zentralisierte Bearbeitung nicht geeignet sind, werden im Auswahlangebot für die Komposition nicht mehr bereitgestellt. Für Anwendungselemente, die eine Baureihe zur Implementierung sowohl einer zentralen als auch einer dezentralen Lösung besitzen, wird die Baureihendifferenzierung während der Implementierung durchgeführt.

Ad 5) Bedeutung von Nullzuständen:

Als Nullzustände werden hier Merkmalausprägungen bezeichnet, die bedeuten, dass ein zugeordnetes Anwendungselement keine Eigenschaften dieses Merkmals besitzt. Die Einführung eines Nullzustandes für ein Merkmal ist davon abhängig, ob es Anwendungselemente geben kann bzw. darf, die diesem Merkmal nicht zugeordnet werden. Werden keine Nullzustandsausprägungen verwendet, so kann nicht mehr unterschieden werden, ob die beschriebenen Anwendungselemente, die einem Merkmal nicht zugeordnet sind, nur noch nicht vollständig beschrieben sind oder, ob sie definitiv keine Eigenschaft dieses Merkmals besitzen.

Darüber hinaus können Beziehungen zwischen Merkmalausprägungen einer Sachmerkmalleiste nur zwischen existenten Ausprägungen gebildet werden.

Für die Merkmalausprägungen von **Kundenbestellung** gilt, dass es eine semantische Beziehung zwischen *Nicht Bestandsgeführt* und *Keine* (*Produktbereitstellung*) gibt. Der Nullzustand in *Produktbereitstellung* ist im ursprünglichen Beispiel jedoch nicht existent, dadurch entsteht ein Konflikt, der durch Einführung des Nullzustandes gelöst werden kann.

3.4.3 Konstruktionskatalog mit polyhierarchischer Gliederung

Will man einen Konstruktionskatalog oder genauer einen Lösungskatalog (siehe Abschnitt 2.1.2.4) für die Konstruktion von Anwendungssystemen aufbauen, so sind die Katalogelemente „Gliederung“, „Hauptteil“ und „Zugriffsmerkmale“ zu erstellen.

Der Hauptteil entspricht den Anwendungselementen und ihren Beschreibungsinformationen. Zugriffsmerkmale werden über die Sachmerkmalleisten abgebildet. Es fehlt also nur noch die Gliederung.

Die Gliederung ist einerseits das Ergebnis der Anwendungsbereichszerlegung bei der Neu- und Erweiterungskonstruktion des Anwendungselemente-Baukastensystems. Andererseits ist die Gliederung ein sehr wichtiges Instrument für den Composer zur Navigation im Komponentenbestand während des Variantenkonstruktionsprozesses.

Wie in Abschnitt 2.1.2.2 dargestellt, gibt es die zwei Relationstypen „Abstraktion“ und „Komposition“ für den Aufbau von Hierarchien. Im Gegensatz zu Konstruktionskatalogen der technischen Konstruktion, ist es für die Komposition von Anwendungssystemen sinnvoll, eine Gliederung aufzubauen, die beide Relationstypen verwendet. Zudem sollte die Gliederung polyhierarchischer Natur sein, d.h. Elemente der Gliederung können mehreren übergeordneten Elementen zugeordnet sein. Dadurch kann ein Begriff der Abstraktionshierarchie gleichzeitig einem oder mehreren Baugruppenbegriffen der Kompositionshierarchie zugeordnet sein. Folglich ist eine ausdrückliche Kennzeichnung der Relationsbeziehung zwischen den Gliederungselementen oder sogar eine geteilte Darstellung (nur auf Präsentationsebene) von zwei verschiedenen Gliederungshierarchien, die jedoch dieselben Elemente verwenden, empfehlenswert.

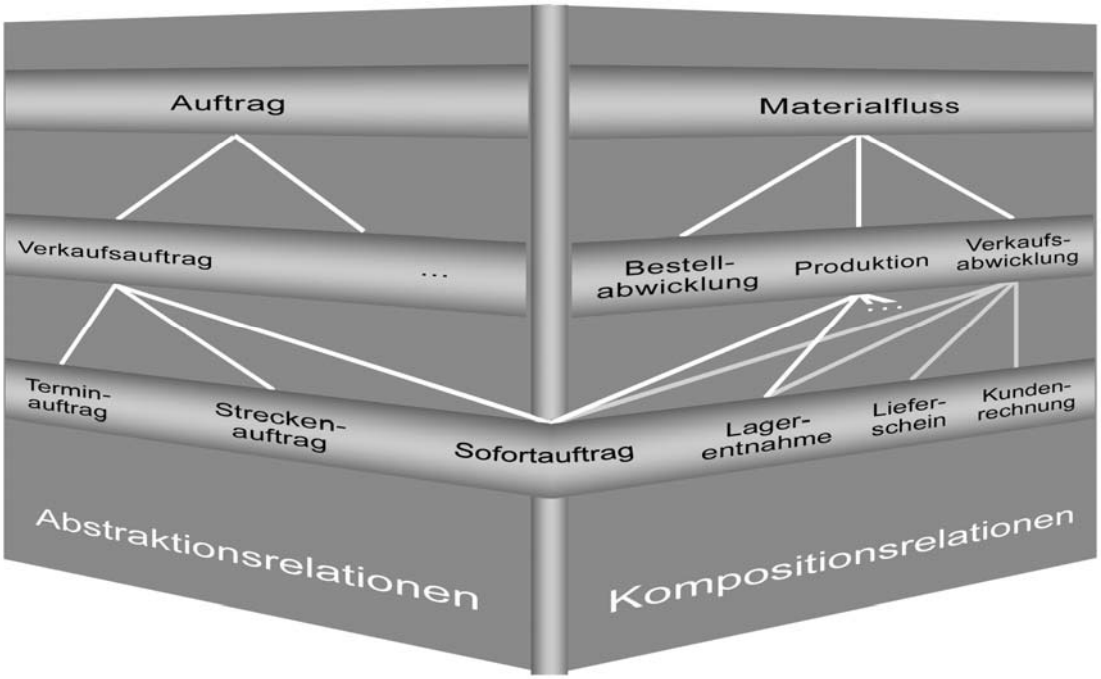


Abbildung 30: Beispiel für eine polyhierarchische Gliederungsstruktur

Jeder Knoten (Begriff) der Gliederung kann eine Sachmerkmalleiste sein. Ihre Existenzbedingung ist einfach die Festlegung von Merkmalen und Ausprägungen für den Begriff und die Zuordnung von Anwendungselementen.

Abstraktionsrelationen unterstützen die Suche nach den richtigen Begriffen im Konstruktionsprozess. Diese Begriffe sind dann die Teilaufgaben, für die geeignete Lösungen gesucht werden müssen. Teilaufgaben entsprechen somit den Sachmerkmalleisten. Kompositionsrelationen stellen in diesem Zusammenhang Referenzmodelle für Komponentenkompositionen oder anders ausgedrückt „optionale Baugruppen“ dar.

Wie in Abschnitt 3.3.7 schon erwähnt, sind die Bauteile eines Anwendungselemente-Baukastensystems nicht immer elementare Anwendungselemente, sondern auch Baugruppen aus komponierten Anwendungselementen. Eine Unterscheidung zwischen Elementen und Baugruppen ist im Ordnungssystem der Anwendungselemente nicht erforderlich[39]. Denn es ist eine Frage der Abgrenzung der Anwendungsdomäne, für welche Abstraktionsebene sich ein Composer noch interessiert, und es ist eine Frage

[39] Aus Sicht der Variantenkonstruktion.

des Entwicklungsaufwandes und der Erfahrung, welche Variabilitäten in einem System realisiert werden. Folglich werden alle Elemente Anwendungselemente genannt, und es kann davon ausgegangen werden, dass sie immer als Baugruppe aus Anwendungselementen realisiert sein können. Jedes elementare Anwendungselement ist unter Umständen nur temporär elementar und kann durch Erweiterung der Variabilität wieder in Anwendungselemente zerlegt werden. Für die Komposition stellt der Elementarzustand nur eine vorübergehende Einschränkung der Variabilität dar und kann über Entwicklungsanforderungen korrigiert werden. Das Gesamtkonzept für das Ordnungssystem ist inkrementell angelegt, so dass sich im Zeitverlauf der Abstraktionsfokus verändern und die Variabilität erhöhen kann. Probleme entstehen nur dann, wenn eine Erhöhung der Variabilität im Fachkonzept eine völlige Veränderung der Realisierung nach sich zieht.

Die Gliederung ermöglicht dem Composer, sich in verschiedenen Abstraktionsebenen[40] gleichzeitig zu bewegen. In einem Anwendungsbereich kann er vorgedachte Baugruppen übernehmen. Bei der nächsten Teilaufgabe ist er daran interessiert, die Eigenschaften seines Anwendungselementes selbst zu komponieren und so seine individuelle Baugruppe flexibel zusammenzustellen.

Diese Baugruppe wurde zwar für ein spezielles Projekt erstellt, sie kann jedoch bei guter Annahme durch den Kunden als Referenz-Baugruppe in das Ordnungssystem aufgenommen werden. Damit sind ihre Implementierung, ihr Fachkonzept sowie ihre Dokumentation über das Ordnungssystem verfügbar. Sachmerkmalleisten für Baugruppen erhalten wiederum nur die Merkmale, die für den Auswahlprozess des Composers von Interesse sind.

Zusammengefasst kann man sagen, dass die Option des Composers, nur Baugruppen zu betrachten und nicht ihre Bestandteile verstehen zu müssen, eine Möglichkeit bietet, die hohe Komplexität eines Anwendungselemente-Baukastensystems zu verbergen.

Für die individuelle Baugruppen-Komposition ist umso mehr Spezialwissen nötig, je detaillierter die Partitionierung der Komponenten fortschreitet. Im Umkehrschluss ist eine umfangreichere Dokumentation für das Verständnis von Baugruppen notwendig, je mehr Anwendungselemente dafür komponiert wurden.

In den folgenden Grafiken sind die Strukturelemente des Ordnungssystems und ihre strukturellen Beziehungen abstrakt im Überblick dargestellt.

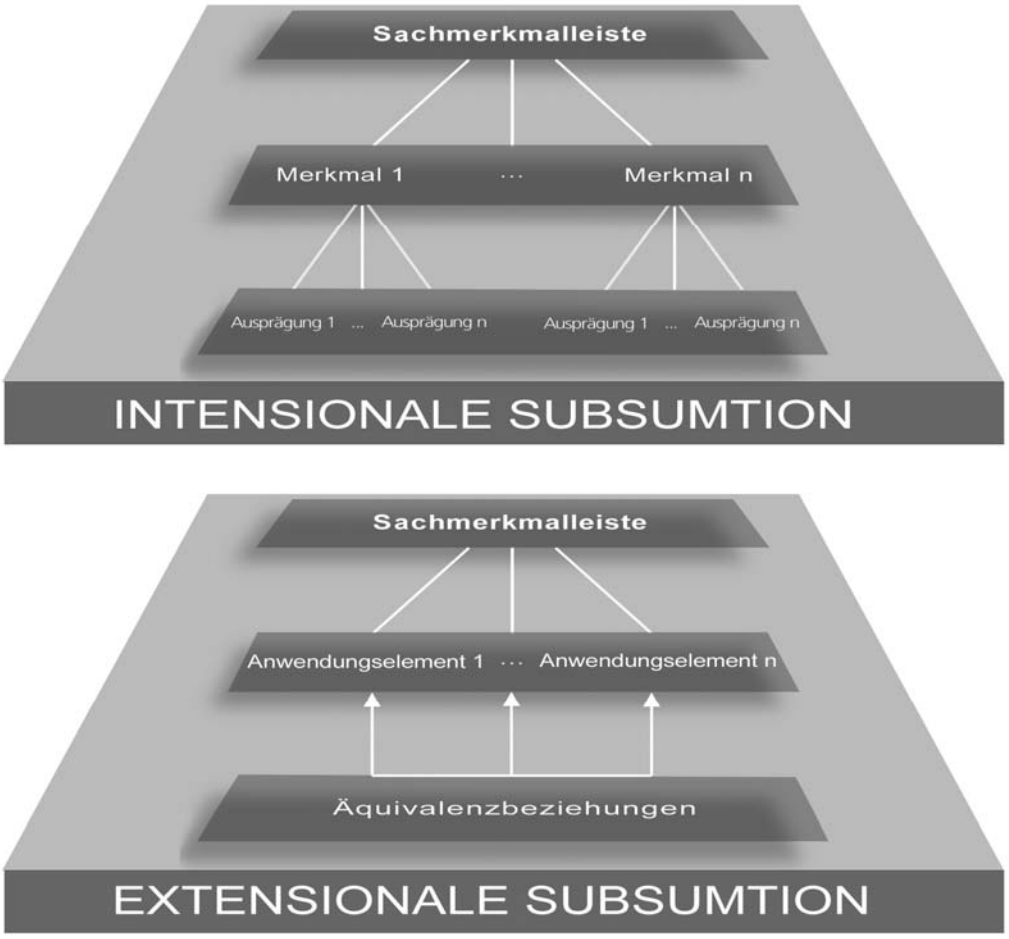


Abbildung 31: Abstraktionsbeziehungen außerhalb der Gliederung

[40] Abstraktionsebene lässt sich hier gut als Verständnis- bzw. Interessensebene interpretieren.

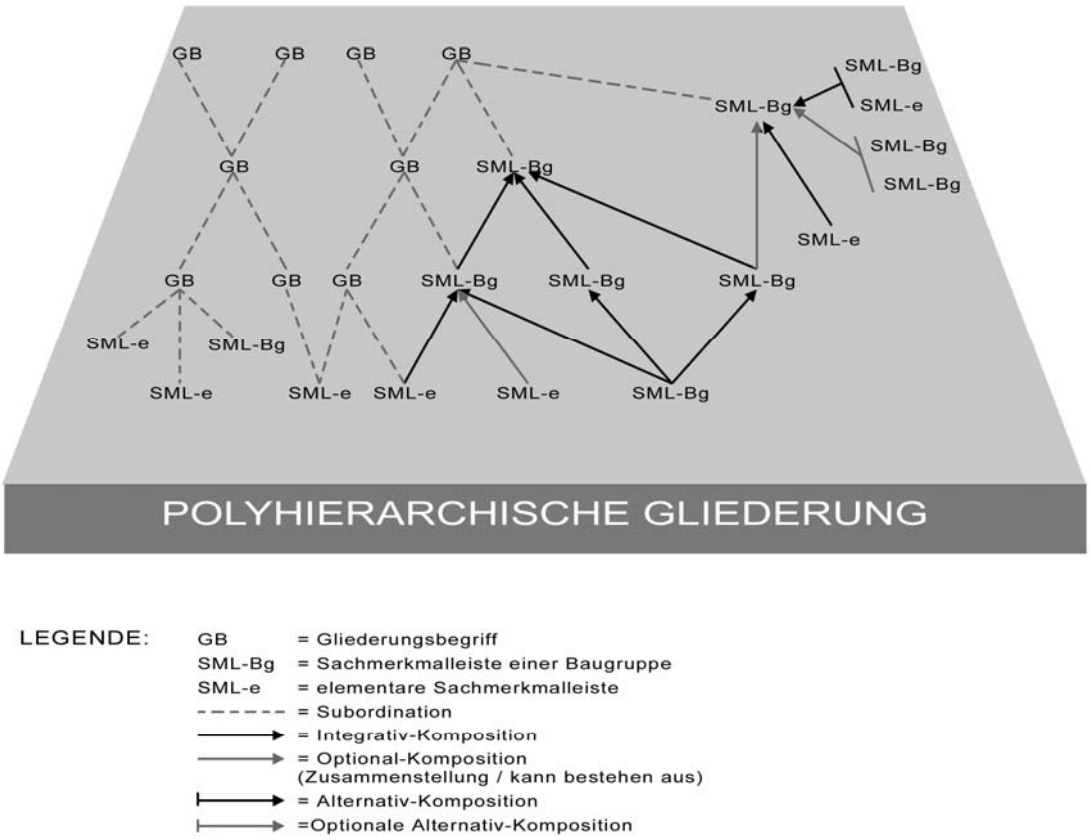


Abbildung 32: Strukturelemente und strukturelle Beziehungen innerhalb der polyhierarchischen Gliederung

Eine Unterscheidung zwischen SML-elementar und SML-Baugruppe ist nur für das bessere Verständnis der Gliederungsstruktur verdeutlicht worden. Die Aussagen dieser Grafik stellen die Zusammenfassung der Strukturierungsmöglichkeiten der Gliederung dar:

- Jede Sachmerkmalleiste stellt einen Gliederungsbegriff dar.
- Aus Sicht des Konstruktionskataloges repräsentiert eine Sachmerkmalleiste ein Varianten-Set von Anwendungselementen (oder Baugruppen).
- Jede neu zusammengestellte Baugruppe kann zur Bildung einer Sachmerkmalleiste führen und wird dann in die Gliederung mit aufgenommen. Die semantischen Relationen zwischen den Elementen

und der Baugruppe selbst sowie ihrer Varianten begründen die strukturellen Kompositionsbeziehungen im Katalog.

- Baugruppen können aus Elementen und aus Baugruppen bestehen.
- Mehrere Sachmerkmalleisten können unter denselben Gliederungsbegriff fallen.
- Die Konkretion in der Abstraktionshierarchie führt, falls im Konstruktionskatalog eine Lösung existiert, zu einer Sachmerkmalleiste. Eine weitergehende Konkretion ist nicht mehr innerhalb der Gliederung möglich. Innerhalb der Gliederungshierarchie kann von einer Sachmerkmalleiste aus nur noch durch Partition fortgeschritten werden.

Strukturelle Kompositionsbeziehungen

Jede Baugruppe kann integrativ-kompositionale und optional-kompositionale (An-)Teile besitzen. Integrativ-Komposition bedeutet, dass auf jeden Fall ein Anwendungselement der Teil-Sachmerkmalleiste als Bestandteil der Baugruppe verwendet werden muss. Die Optional-Komposition stellt es frei, ein Anwendungselement der Teil-Sachmerkmalleiste für die Baugruppen-Komposition zu verwenden. Für die Alternativ-Komposition und die optionale Alternativ-Komposition gilt analog dasselbe, nur dass dafür zwischen zwei oder mehreren Teil-Sachmerkmalleisten ausgewählt werden kann. Im nächsten Abschnitt werden die semantischen Relationen vorgestellt. Sie repräsentieren die Abhängigkeitsbeziehungen zwischen den Elementen des Ordnungssystems. Beziehungen auf semantischer Ebene sind die Vorgaben für die strukturellen Beziehungen zwischen abstrakten Elementen im Konstruktionskatalog.

Baugruppen, die nicht zur Bildung einer Sachmerkmalleiste führten, sind im Konstruktionskatalog nicht vorhanden. Sie können trotzdem im Repository für die Komponentenverwaltung existieren und für zukünftige Konstruktionsprojekte verwendet werden.

Baugruppenbeschreibung - Eigenschaftspropagierung

Werden Baugruppen gebildet, so entstehen neue Gegenstände derselben Sprachebene wie ihre Teile [Schienmann97, 59]. Der Grund für die Komposition von Baugruppen ist, dass ein neuer Gegenstand mit eigenständigen Eigenschaften nicht

ausschließlich aus den Eigenschaften seiner Teile, gebildet wird. Natürlich bedingen die Eigenschaften der zusammengesetzten Bestandteile die Eigenschaften des Ganzen.

Im Bereich des Software-Konfigurations-Managements werden Eigenschaften durch Propagierung von Elementen an ihre Baugruppen[41] weitergereicht. Bei Zeller und Snelting werden hierbei die Eigenschaften der Elemente durch Intersektion reduziert [Zeller/Snelting97, 13]. Sind jedoch so genannte „unabhängige“ Eigenschaften für die Elemente vorhanden, so werden diese bei der Propagierung beibehalten (z.B. Autor). Diese Vorgehensweise funktioniert unter der Voraussetzung, dass nur Elemente zu Baugruppen konfiguriert werden können, die identische Merkmale von abhängigen Eigenschaften aufweisen. Zeller und Snelting wollen damit auch gleich eine Konsistenzprüfung für die Baugruppenbildung durchführen. Die ***Methode der semantischen Komposition*** lässt nur mittelbar, über die Indikation von Abhängigkeiten durch nicht-orthogonale Eigenschaften, Rückschlüsse der Eigenschaftsbeschreibung auf die Kombinierbarkeit von Anwendungselementen zu. Für eine Eigenschaftspropagierung müsste eine Vereinigung aller Eigenschaften und anschließender Entfernung von Dubletten abhängiger Eigenschaften durchgeführt werden. Die Frage, die sich dabei wie auch bei Zeller und Snelting stellt: Welches sind abhängige und unabhängige Eigenschaften?

Die ***Methode der semantischen Komposition*** schlägt eine andere Vorgehensweise vor. Baugruppen werden wie Anwendungselemente mit denjenigen Eigenschaften beschrieben, die auf der aktuellen Abstraktionsebene für einen Composer von Interesse sind.

Daraus lassen sich drei Aussagen ableiten, die für die Eigenschaftsbeschreibung von Baugruppen gelten:

1. Es sollen nur diejenigen Eigenschaften genannt werden, die den Auswahlprozess unterstützen.
2. Zur Verringerung der Informationsmenge können auf Baugruppenebene Komplexmerkmale ursprüngliche Einzelmerkmale der Bauteile bündeln.

3. Für die Zusammenstellung von Merkmalen und Ausprägungen für eine Baugruppen-Sachmerkmalleiste stehen die „sinnvollen“ Kompositions-Varianten im Vordergrund.

Werden nun im Zuge der Variantenkonstruktion neue Baugruppen komponiert, die ebenfalls unter eine existierende Sachmerkmalleiste fallen (Sortimentserweiterung auf Baugruppenebene), so werden ihre Merkmale und Ausprägungen im Bedarfsfall entsprechend erweitert.

Merkmalarten und Merkmalbestimmung

Es ist gerade eine Stärke von Merkmalbeschreibungssystemen und im Speziellen von Sachmerkmalleisten, dass sie verschiedene Sichten auf eine Fragestellung bzw. Teilaufgabe umfassen können. So können unterschiedliche Verwendergruppen von Sachmerkmalleisten sich auf die Merkmale ihres Interesses konzentrieren. Daraufhin sollten einige Aspekte der Merkmale ausführlicher betrachtet werden. Meinl [Meinl90, 7ff] hat eine ausführliche Übersicht der verschiedenen Merkmalarten erstellt. Er zeigt damit, welche Eigenschaften in einer Sachmerkmalleiste beschrieben werden können. Koller und Kastrup haben eine Einteilung der Merkmale von Lösungskatalogen vorgenommen [Koller/Kastrup94, 31f]:

1. systematisierende Merkmale (Gliederungsmerkmale)
2. Beschreibungsmerkmale
3. auswahlerleichternde Merkmale (Zugriffsmerkmale)

Maßgebend für diese Dreiteilung ist die Struktur eines Konstruktionskataloges.

Die ***Methode der semantischen Komposition*** definiert für die Sicht des Composers auf Sachmerkmalleisten die Merkmale vom Typ „Zugriffsmerkmale“ als wesentlich. Gliederungsmerkmale werden in einer separaten Struktur - der Gliederung des Konstruktionskataloges - verwaltet. Beschreibungsmerkmale sind für den Composer als Zusatzinformation zu verwenden. Für die Herstellung von neuen Anwendungs-

[41] Im Original: feature propagation from components to configurations [Zeller/Snelting97, 28].

elementen stellen sie jedoch essentielle Anforderungen dar (Anforderungsanalyse bzw. Spezifikation).

Die Zuordnung einer Eigenschaft eines Anwendungselementes zu einer der drei Merkmalkategorien ist jedoch erstens nicht trivial und zweitens nicht (dauerhaft) eindeutig.

Auf der einen Abstraktionsebene kann ein Merkmal als Gliederungsmerkmal verwendet werden, auf einer höheren Abstraktionsebene wird derselbe Begriff als Zugriffsmerkmal eingesetzt (Baugruppenebene). Auf diese Verschiebung der Begriffe im Ordnungssystem wird im Abschnitt 3.4.6 noch einmal kurz eingegangen.

Damit soll deutlich gemacht werden, dass eine klare Trennung zwischen Gliederung und Sachmerkmalleisten sowie zwischen auswahlerleichternden Merkmalen und beschreibenden Merkmalen nicht möglich ist. Nicht einmal zwischen Ausprägungen und Merkmalen kann dauerhaft eindeutig differenziert werden – was in einem bestimmten Zusammenhang besser eine Ausprägung ist, kann aus anderer Sicht ein Merkmal darstellen.

Katalog-Extrakte

Das Ordnungssystem für Anwendungselemente muss unter verschiedenen Perspektiven betrachtet werden. Es gibt ein Ordnungssystem für das gesamte Angebot an Anwendungselementen im Baukastensystem. Der Anwendungsbereich, der hierfür zugrunde liegt, muss die Vielfalt einer gesamten Branche berücksichtigen.

Steht ein konkretes Konstruktionsprojekt an, so wird mit nur wenigen Handlungen (Entscheidungen über Unternehmensstruktur, Produktspektrum und Zielkunden) daraus ein „angepasstes“ Angebot erzeugt, indem das Sortiment der Anwendungs-elemente reduziert wird. Diese Reduzierung wird jedoch nicht nur durch Einschrän-kungen innerhalb von Sachmerkmalleisten realisiert, sondern auch durch eine Reduzierung der Gliederung und somit des Ordnungssystems. Angepasstes Angebot bedeutet also auch, dass nur eine Teilmenge des Ordnungssystems für ein Anwenderunternehmen sichtbar ist (zielgruppenorientierter Katalog).

Ist ein Anwendungssystem fertig implementiert, so muss das Anwenderunternehmen eine Abbildung seiner Installation besitzen. Das Instrument für die Abbildung des

Anwendungsbauplans der installierten Anwendungselemente ist eine „Instanz“ des Ordnungssystems für Anwendungselemente. Inhaltlich ist es reduziert auf die ausgewählten Anwendungslösungen und deren strukturelle Beziehungen. Für eine vollständige Systemdokumentation müssen außer dem Anwendungsbauplan alle weiteren Vereinbarungen und Konstruktionsentscheidungen in elektronischer Form vorhanden sein (z.B. Workflow- und Prozesskonfiguration, Modellierung der Unternehmensorganisation, Konfiguration der benutzerspezifischen Arbeitsplätze und weitere).

Dem Anwenderunternehmen stehen also zwei Kataloge zur Verfügung. Der Anbieter-Katalog des Herstellers, aus dem neue Angebote für Erweiterung und Umbau der installierten Anwendung ausgewählt werden können und der interne Katalog, der den Anwendungsbauplan für die bereits installierte Anwendung enthält. Beide Kataloge müssen für Erweiterungen der bestehenden Anwendung interagieren, denn im Anbieter-Katalog muss ein auf die bisher getroffenen Konstruktionsentscheidungen abgestimmtes Angebot enthalten sein.

3.4.4 Semantische Relationen zwischen Anwendungselementen

Beziehungen zwischen Anwendungselementen unterliegen ebenso wie die Begriffsbeziehungen einer Sprache semantischen Relationen, welche die semantische Struktur des Wortschatzes (in diesem Falle die Wechselwirkungen zwischen Anwendungselementen) darstellen [Bußmann90].

Die Beziehungen zwischen den Anwendungselementen können im Wesentlichen durch die logischen Operationen der Äquivalenz, Implikation und Negation beschrieben werden. Die Orthogonalität als so genannte Nicht-Beziehung wird als Standardbeziehung unterstellt und im System nicht erfasst.

Zwei Anwendungselemente sind solange **orthogonal** zueinander, bis eine semantische Relation diese Unabhängigkeit einschränkt.[42]

[42] Der Grad der Orthogonalität der Eigenschaften zweier Anwendungselemente (siehe Abschnitt 3.3.4.2) hat nur Indikatorwirkung. Ihre „wirkliche“ Unabhängigkeit bezüglich ihres Einsatzes in einem Anwendungssystem wird durch ihre semantischen Relationen bestimmt.

Äquivalenzbeziehungen repräsentieren „vollständige Austauschbarkeit“ von Anwendungselementen und werden durch die Anwendungselemente-Varianten, die in einer Sachmerkmalleiste zusammengefasst sind, im System abgebildet. Implikation und Negation werden explizit im System erfasst. Echte Wechselwirkungen, die auch Konflikte in der Komposition ergeben können, existieren nur zwischen Anwendungselementen, die in einer Implikations- oder Negationsbeziehung zueinander stehen.

Beispielsweise kann Anwendungselement X nicht mit Anwendungselement Y zusammenarbeiten (Inkompatibilität) oder Anwendungselement A kann nicht ohne Anwendungselement B seine Aufgabe erfüllen (informative Vollständigkeit). Zunächst macht es keinen Unterschied, ob diese Wechselwirkungen technischer oder fachlicher Natur sind. Ein Composer-Werkzeug muss diese Beziehungen verwalten und während des Kompositionsvorganges permanent anbieten, damit die Auswahl der Anwendungselemente konsistent durchgeführt werden kann.

Beziehungsart	**Darstellung**	**Bedeutung**
Implikation	$A \rightarrow B$	Informative Vollständigkeit
Negation	$A \wedge \neg B$	Inkompatibilität
Äquivalenz	$A \leftrightarrow B$	Austauschbarkeit
Orthogonalität	$A \perp B$	Unabhängigkeit

Tabelle 8: Beziehungsarten der semantischen Relationen

Beispiele:

($\rightarrow$) Eine Kundenbestellung für Lagerartikel benötigt in jedem Fall eine Rahmenvereinbarung.

($\neg$) Ein Streckengeschäft darf keine Lagerentnahme enthalten.

($\leftrightarrow$) Firmenkundenrechnungen und Privatkundenrechnungen sind in jeder möglichen Verwendung vollständig austauschbar.

($\perp$) Die Auswahl der Form für die Rechnungsadresse hat keine Auswirkung auf Festlegungen bezüglich der Lieferadressen.

Die grau hinterlegten Wortgruppen stellen so genannte Schlüsselworte für das Erkennen von semantischen Relationen dar. Aussagen über fachliche Zusammenhänge eines Anwendungsbereichs müssen während der Aufbauphase eines semantischen Beziehungsnetzes und während seines Einsatzes laufend diesbezüglich geprüft werden.

Die Eigenschaften der semantischen Relationen lassen sich in folgender Übersicht darstellen:

Beziehungsart	Eigenschaften der semantischen Relationen
Implikation	Transitiv
Negation	irreflexiv, symmetrisch
Äquivalenz	reflexiv, symmetrisch, transitiv
Orthogonalität	reflexiv, symmetrisch

Tabelle 9: Eigenschaften der semantischen Relationen

Die Eigenschaften der semantischen Relationen verdeutlichen, was eigentlich offensichtlich ist. Die Implikation erzeugt unter Umständen unangenehme Verkettungen; sie gilt in der Regel nur in eine Richtung. Die Negation bedeutet ein sich gegenseitiges Ausschließen zwischen Anwendungselemente-Paaren. Eine Transitivität liegt nicht vor. Bei der Äquivalenzrelation ist die Transitivität eine gewünschte Eigenschaft. Dadurch soll die Freiheit der Variantenwahl ausgedrückt werden.

Die Äquivalenzrelation zwischen Anwendungselementen wird im Ordnungssystem durch die Zuordnung zu derselben Sachmerkmalleiste ausgedrückt. Allerdings kann die Austauschbarkeit leicht durch die gleichzeitige Verwendung von Implikation und Negation eingeschränkt werden, so dass eine Situation entsteht, bei der keine der Anwendungselemente-Varianten gegen eine andere ersetzt werden kann, ohne dass auch andere schon ausgewählte Anwendungselemente ebenfalls ausgetauscht werden müssen. Anwendungselemente-Varianten einer Sachmerkmalleiste können dadurch ihre Symmetrieeigenschaft und/oder ihre Transitivitätseigenschaft verlieren. Dann sind sie jedoch nicht mehr äquivalent.

Bei der Orthogonalität ist zu beachten, dass sie nicht transitiv ist. Die Symmetrieeigenschaft ist durch die Definition der paarweisen Prüfung vorgegeben.

Eine weitere Eigenschaft der semantischen Relationen ist von Interesse: ihre paarweise Disjunktion. Sie wird allerdings nur für die Implikation und die Negation als notwendige Bedingung festgelegt. Zum einen ist dies dadurch begründet, dass nur diese beiden Relationen ausdrücklich im System definiert werden und zum anderen kann die Disjunktion nicht für alle Fälle bindend festgelegt werden. Beispielsweise kommt es vor, dass eine Äquivalenz- und eine Negationsbeziehung oder eine Äquivalenz- und eine Orthogonalitätsbeziehung zwischen zwei Anwendungselementen zeitgleich bestehen. Eine Prüfung der Disjunktion von Implikation und Negation lässt sich sehr leicht durchführen und kann dadurch eine Fehlerquelle bei der Beziehungsdefinition beseitigen.

Die Beziehung, die den höchsten Aufwand für ihre Verwaltung und Berücksichtigung bei der Komposition verursacht, ist die Implikation. Sie stellt zugleich das wichtigste Hilfsmittel für die Beherrschung der Kombinationsproblematik der ***Methode der semantischen Komposition*** dar und wird im Folgenden näher betrachtet.

Eine Implikationsbeziehung kann für eine aufgrund technischer Bedingungen nicht erreichte Orthogonalität als Notausgang verwendet werden. Von Creatorseite muss dann eine Implikationsbeziehung erfasst werden. Diese Implikationsbeziehung kann als „Implementierungsabhängigkeit“ gekennzeichnet werden (z.B. Implikation/I, $A \overset{i}{\rightarrow} B$)

Die „informative Vollständigkeit“ ist auf der fachlichen Ebene für die Integration von Anwendungen zuständig. Dieser Aspekt der semantischen Relation hat einen sehr weiten Wirkungsbereich. Die Implikation kann Anwendungselemente für eine geeignete Informationserfassung und –verwertung innerhalb des Aufgabenfeldes eines Anwenders oder im Bereich einer Abteilung verknüpfen. Es können jedoch auch unternehmensbereichsübergreifende Aufgaben existieren, die miteinander kooperieren müssen. Dabei wird deultich, dass einige der Implikationsbeziehungen für einen Composer klar ersichtlich und verständlich sind. Andere Implikationen hat er u. U. nicht in seinem Kompositionsfokus, da sie andere Arbeitsbereiche betreffen. Entscheidungen dort, können Auswirkungen auf seine Kompositionsmöglichkeiten haben.

Implikationsbeziehungen stellen einerseits Einschränkungen der Kombinationsfähigkeit von Anwendungselementen dar. Andererseits führen sie den Composer durch den Konstruktionsprozess, indem sie auf noch ausstehende Teilaufgaben und Anwendungselemente hinweisen. Demgegenüber sind Negationsbeziehungen reine Kombinationsverhinderer. Äquivalenzbeziehungen stellen ein Angebot an Alternativen

dar. Die Orthogonalität, neben der Äquivalenz die anzustrebende Beziehung zwischen Anwendungselementen, ist nicht vollständig zu erreichen.

Damit die Implikation den Kompositionsprozess flexibler unterstützen und dadurch die Realität des Anwendungsbereiches besser abbilden kann, werden Implikationsbeziehungen differenziert betrachtet. Es wird unterschieden in:

Beziehungsart	Darstellung	Schlüsselworte
Implikation	A $\rightarrow$ B	benötigt in jedem Fall
Alternativ-Implikation	A $*\rightarrow$ B	benötigt entweder ... oder
Optional-Implikation	A $\circ\rightarrow$ B	benötigt eventuell
Optionale Alternativ-Implikation	A $\circ*\rightarrow$ B	benötigt eventuell entweder ... oder

Tabelle 10: Übersicht der Implikationsbeziehungen

Erfassung der semantischen Relationen

Es ist zu aufwändig und daher nicht sinnvoll, alle Beziehungen zwischen Anwendungselemente-Exemplaren zu bestimmen und im System zu hinterlegen. Durch das hier vorgestellte Ordnungssystem für Anwendungselemente ist es möglich, neben den Anwendungselementen auch Sachmerkmalleisten, Merkmale und Merkmalausprägungen für den Aufbau eines Beziehungsnetzes zu verwenden. Ausgewählte Gliederungsbegriffe, die eine betriebswirtschaftlich herausragende Gruppierung von Lösungsangeboten repräsentieren (beispielsweise alle Baugruppen für einen Geschäftsprozess) werden als Kategorie bezeichnet und stellen ebenfalls wichtige Aggregate für die Beziehungsdefinition dar.

Die Rangordnung der Mächtigkeit der Beziehungsauswirkungen entspricht der Anordnung in folgender Matrix.

A **R** B A B	Anwendungs-element	Merkmal-ausprägung	Merkmal	Sachmerk-malleiste	Kategorie	
Anwendungs-element						
Merkmal-ausprägung						
Merkmal						
Sachmerk-malleiste						
Kategorie						

Tabelle 11: Matrix der möglichen Beziehungselemente-Kombinationen im Ordnungssystem

Wäre jede dieser Beziehungselemente-Kombinationen geeignet für den Einsatz mit allen Beziehungsarten und würde dann noch berücksichtigt, dass die Beziehungselemente auf unterschiedlichen Abstraktionsebenen verwendet werden können, so ließe sich der Aufwand zur korrekten Erfassung aller Beziehungen nicht mehr beherrschen.

Hinzu kommt noch, dass diese Beziehungen in der Variantenkonstruktionsphase auch auf ihre Erfüllung überprüft werden müssen, d.h. für die Benutzer-Interaktion des Variantenkonstrukteurs muss die Systemantwortzeit für die „Berechnung“ der vorgeschlagenen und zulässigen Lösungskombinationen in einem zumutbaren Bereich liegen.

Die nächste Tabelle zeigt, wie der Beziehungsarteneinsatz für bestimmte Beziehungs-elemente-Kombinationen eingeschränkt werden kann.

A **R** B A B	Anwendungs-element	Merkmal-ausprägung	Merkmal	Sachmerk-malleiste	Kategorie	
Anwendungs-element	$A \rightarrow B,\ A \leftrightarrow B$ $A \wedge \neg B, A \perp B$	—	—	$A \rightarrow B$ $A \wedge \neg B, A \perp B$	$A \rightarrow B$ $A \wedge \neg B, A \perp B$	
Merkmal-ausprägung	$A \rightarrow B$ $A \wedge \neg B$	$A \rightarrow B$ $A \wedge \neg B$	$A \rightarrow B$ $A \wedge \neg B$	$A \rightarrow B$ $A \wedge \neg B$	—	
Merkmal	$A \rightarrow B$ $A \wedge \neg B$	$A \rightarrow B$ $A \wedge \neg B$	$A \rightarrow B$ $A \wedge \neg B$	$A \rightarrow B$ $A \wedge \neg B$	—	
Sachmerk-malleiste	$A \rightarrow B$ $A \wedge \neg B, A \perp B$	—	—	$A \rightarrow B,\ A \leftrightarrow B$ $A \wedge \neg B, A \perp B$	$A \rightarrow B$ $A \wedge \neg B, A \perp B$	
Kategorie	$A \rightarrow B$ $A \wedge \neg B, A \perp B$	—	—	$A \rightarrow B$ $A \wedge \neg B, A \perp B$	$A \rightarrow B,\ A \leftrightarrow B$ $A \wedge \neg B, A \perp B$	

Tabelle 12: Matrix des eingeschränkten Beziehungsarteneinsatzes in den Beziehungselemente- Kombinationen

Trotzdem wäre die Verwaltung der semantischen Relationen unbeherrschbar, wenn alle Möglichkeiten, die hier angeboten werden, bedingungslos genutzt würden.

Es werden im Folgenden einige Regeln für das Beziehungsdesign vorgeschlagen, die bei der Vermeidung eines Beziehungschaos unterstützend mitwirken sollen. Die vorgestellten Regeln können nur Richtlinien darstellen, denn ihre absolute Gültigkeit ist sehr stark abhängig vom konkreten Anwendungsbereich und von dem Werkzeug, welches zur Beziehungsverwaltung eingesetzt wird.

Regel 1: Möglichst nur semantische Relationen zwischen Beziehungs-elementen auf derselben Abstraktionsebene oder einer Ebene darüber oder darunter definieren.

Regel 2: Ist Regel 1 nicht einzuhalten, dann sollten semantische Relationen möglichst nur innerhalb einer Kompositionshierarchie definiert werden.

Regel 3: Wenn möglich, sollten immer semantische Relationen zwischen den von der Beziehungsauswirkung her mächtigsten Elemente definiert werden.

Regel 4: Bei gleichwertigen Lösungen (mehrere Beziehungskombinationen können dieselbe Wirkung besitzen), ist immer die Lösung mit der absolut

geringsten Anzahl an semantischen Relationen zu bevorzugen. Im Normalfall kommt diese Regel nach konsequenter Anwendung der Regel 3 schon nicht mehr zum Einsatz.

Regel 5: Es ist eine maximale Anzahl von Anwendungselemente-Verkettungen durch Implikation festzulegen. Die permanente Prüfung dieser Grenze ist als Frühwarnsystem zu implementieren.

Existieren fachlich begründete Beziehungen zwischen Anwendungselementen, die aufgrund der Beschränkungen der hier vorgestellten Regeln nicht korrekt im Beziehungsnetz abgebildet werden könnten, so gibt es zwei Lösungsmöglichkeiten:

1. Die Regeln für das Beziehungsdesign wurden nicht ausreichend an den Anwendungsbereich angepasst. Dies muss dann nachgeholt werden. Gibt es dabei Probleme mit dem Verwaltungswerkzeug, so ist eventuell auch die Entscheidung für den Einsatz des vorliegenden Werkzeugs zu prüfen.
2. Anwendungselemente müssen „re-designed“ werden. Durch eine Umgestaltung der Anwendungselemente können Abhängigkeitsbeziehungen aufgelöst werden (Erhöhung der Orthogonalität).

3.4.5 Wissensbasiertes Konfigurieren

*Um **Wissen** zu erlangen,*

***füge** jeden Tag etwas **hinzu**;*

*um **Weisheit** zu erlangen,*

***nehme** jeden Tag etwas **weg**.*

Lao-Tzu in Tao Te Ching

Die Komplexität des Komponenten-Zusammenspiels eines Gesamtsystems wird durch sein Komponenten-Beziehungsnetz ausgedrückt. Darin enthaltene Kompositionsbeziehungen und semantische Relationen sind für die Komponenten-Konfiguration von besonderem Interesse. Es ist offensichtlich, dass ein Beziehungsnetz mit dieser Aufgabenstellung selbst immer komplexer wird. Komplexität von hoch variablen Anwendungssystemen lässt sich nicht wegdefinieren – nur verlagern. Die Verwaltung der Beziehungen zwischen Anwendungselementen führt leicht zu einem Komplexitätsproblem der Kategorie „NP-vollständig“. Darunter sind Probleme zu verstehen, die nicht mehr „effizient lösbar“ sind. „Als nicht effizient lösbar werden alle Probleme angesehen, deren Anstieg an Rechenzeitbedarf mit der Größe der Instanz nicht polynomial beschränkt werden kann, also z.B. Probleme mit exponentiellem Zeitbedarf“ [Nebel95, 7].

Nebel erklärt, dass eine Einschränkung der Wissensrepräsentationssprache auch eine Reduktion des Problemlösungsaufwandes auf polynomialen Zeitbedarf erwirken kann und somit ein akzeptables Antwortzeitverhalten möglich wird. Konkret ist das möglich, wenn für einen Anwendungsbereich ein eigener Wissenrepräsentationsformalismus geschaffen wird, der einen gelungenen Kompromiss zwischen Ausdruckskraft und Berechnungsaufwand darstellt. Die ***Methode der semantischen Komposition*** repräsentiert das Anwendungsbereichswissen für betriebswirtschaftliche Anwendungssysteme durch ihr Ordnungssystem. Die Mächtigkeit der Kompositionsbeziehungen und der semantischen Relationen ist maßgebend für die Ausdruckskraft der Wissensrepräsentation.

Expertensystem für die *Methode der semantischen Komposition*

Für die Verarbeitung komplexer Problemlösungsanforderungen lassen sich Expertensysteme einsetzen. Der grundsätzliche Aufbau eines Expertensystems für die Komposition von Anwendungselementen wird in folgender Grafik gezeigt:

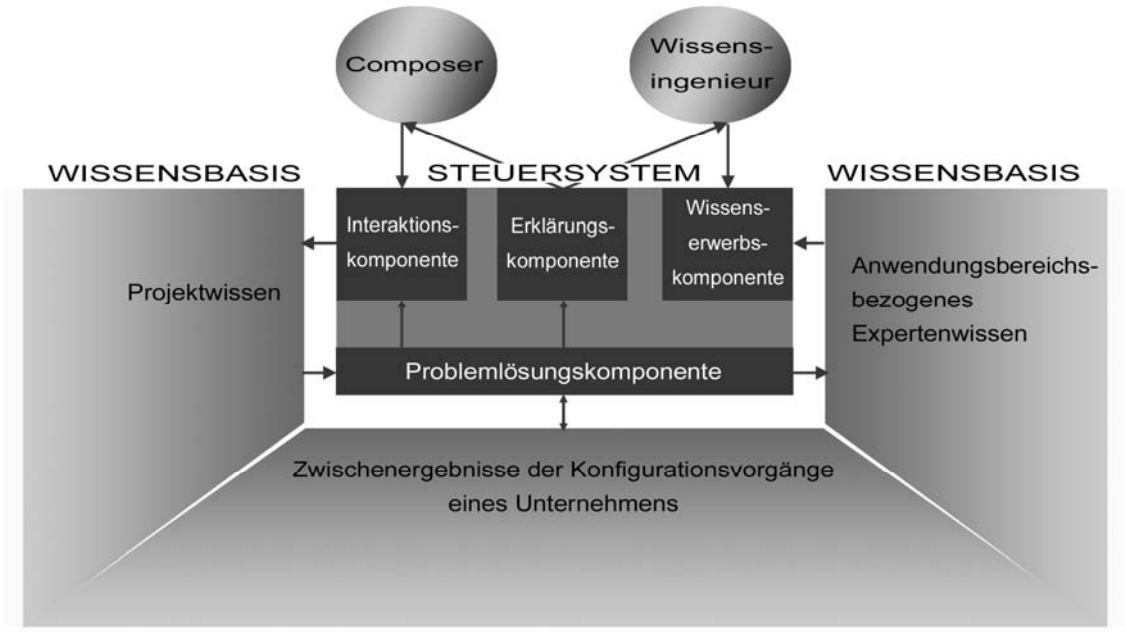


Abbildung 33: Architektur eines Expertensystems für die Anwendungselemente-Konfiguration [vgl. Puppe91, 13]

Im Folgenden wird die architektonische Struktur eines Expertensystems für die Anwendungselemente-Konfiguration erläutert. Ein Expertensystem dieser Art ist geeignet für eine Integration in ein Komponenten-Management-System.

Ein Expertensystem besteht nach Puppe aus den zwei Hauptmodulen Wissensbasis und Steuersystem [Puppe91, 12ff]. Die Wissensbasis besteht aus den Teilen:

- **Projektwissen:** Projektwissen ist das Erfahrungswissen vergangener Projekte (fallbasiertes Wissen). Dieses Wissen wird in Form von Referenz-Baugruppen oder neuen bzw. geänderten semantischen Relationen in der Wissensbasis abgelegt. Projektwissen sollte möglichst mit Vorbedingungen und Begründungen gespeichert werden.

- **Zwischenergebnisse aller Konfigurationsvorgänge:** Alle Zwischenergebnisse von Kompositionshandlungen eines Unternehmens müssen in einem lokalen Repository verwaltet werden. Die hohe Integration betriebswirtschaftlicher Anwendungssysteme und die Forderung nach permanenter flexibler Anpassung an Veränderungen lassen es nicht zu, existente (Teil-)Konfigurationen als vollendet zu betrachten. Werden weitere Kompositionshandlungen durchgeführt, so muss immer eine Prüfung aller abhängigen Komponenten, auch von bereits installierten Komponenten, auf Konfliktfreiheit erfolgen, damit eine neue konsistente Konfiguration entstehen kann.
- **Anwendungsbereichbezogenes Expertenwissen:** Der Konstruktionskatalog und die semantischen Relationen der ***Methode der semantischen Komposition*** bilden das Anwendungsbereichswissen vollständig ab.

Auch das Steuersystem hat verschiedene Untermodule:

- **Interaktionskomponente:** In Expertensystemen wird hier die Schnittstelle hin zum Benutzer des Systems gesehen. Ein Expertensystem für die Unterstützung der Konfiguration von Anwendungselementen wird in ein Komponenten-Management-System eingebettet. Die Interaktion eines Composers mit dem Expertensystem erfolgt also indirekt über das Komponenten-Management-System.
- **Erklärungskomponente:** Jeder Konflikt und jeder Vorschlag, den das Expertensystem seinen Anwendern zur Verfügung stellt, muss begründet werden. Nur so kann die Vorgehensweise des Expertensystems seinen Anwendern transparent werden.
- **Wissenserwerbskomponente:** Neben den Composern gibt es noch eine zweite Anwendergruppe für ein Expertensystem, die Wissensingenieure (Experten). Wissensingenieure bauen die Wissensbasis für den Anwendungsbereich auf. Sie erweitern und korrigieren dieses Wissen in Übereinstimmung mit der Weiterentwicklung des zu Grunde liegenden Baukastensystems für Anwendungselemente. Von den Wissensingenieuren wird dabei eine hohe Leistung gefordert, denn der Aufbau des Beziehungsnetzes im Ordnungssystem (Konstruktionskatalog und semantische Relationen) ist neben der Anwendungs-

elemente-Modellierung die zweite erfolgskritische Aufgabe der ***Methode der semantischen Komposition***. Durch einen häufigen Einsatz des Baukastensystems bei der Herstellung von Standard-Anwendungen fließen die Erfahrungen aus den Konstruktionsprojekten in die Wissensbasis mit ein und verbessern dadurch ihre Güte. Folglich werden die Wissensingenieure durch den inkrementellen Ansatz der Methode entlastet.

- **Problemlösungskomponente:** Für die Interpretation des Experten-wissens zur Lösung des spezifizierten Problems werden „Inferenzmaschinen“, ähnlich einem Programminterpreter oder Compiler in einer Programmierumgebung, eingesetzt.

Die Verwaltung des Anwendungsbereichswissens im Ordnungssystem für Anwendungselemente trägt dazu bei, dass dieses Wissen wirtschaftlich sinnvoll erzeugt wird. Denn das Wissen wird von Branchen-Anwendungsspezialisten zentral einmal erfasst (Aufbau des Konstruktionskatalogs und der semantischen Relationen) und steht dezentral in vielen Konstruktionsprojekten zur Nutzung zur Verfügung (Herstellung von unternehmensindividuellen Anwendungssystem-Ausprägungen). Die Komplexität, die hoch flexiblen Anwendungssystemen inhärent ist, kann damit nicht aufgelöst werden. Sie wird jedoch auf die Erstellung von Konstruktionskatalogen und semantischen Relationen konzentriert.

Forschungsbereich wissensbasiertes Konfigurieren

Die wissensbasierte Konfiguration hat sich in der KI (Künstliche Intelligenz) als einer der wenigen Bereiche mit praktischem Erfolg herausgestellt [Bachmann93]. Im Besonderen erzielte das vom BMBF geförderten Verbundprojekt „PROKON“ erfolgreiche Ergebnisse mit ausgesprochener Praxisrelevanz [Günter95].

Günter [Günter et.al. 94, 95b] charakterisiert die Problemklasse „Konfigurieren“ als Syntheseaufgabe mit folgenden Voraussetzungen:

- **Konfigurationsziele:** Spezifikation der Aufgaben, die angeben, welche Anforderungen die fertige Konfiguration erfüllen soll.
- Eine Menge von Objekten der Anwendungsdomäne und deren Eigen-schaften.

- Eine Menge von Relationen und Restriktionen zwischen den Objekten, insbesondere von kompositionellen Beziehungen.
- **Kontrollwissen:** Wissen über die Vorgehensweise bei der Konfigurierung.

Werden Konfigurationslösungen für Konstruktionsaufgaben im Allgemeinen betrachtet, geht es meist darum, immer wieder ähnliche Produkte zu konstruieren. Außerdem sind überwiegend sehr begrenzte Anwendungsbereiche in den Projekten fokussiert.

Davon verschieden sind die Konstruktionsaufgaben in dem hier vorgestellten Ansatz. Die Neukonstruktion eines Baukastensystems für Standard-Anwendungen wird nur einmal durchgeführt. Der Hersteller eines Baukastensystems gibt das Wissen darüber mit Sicherheit nicht an Dritte weiter. Ein zweites ähnliches Baukastensystem wird deshalb nicht entwickelt. Erweiterungen des Baukastensystems für eine Branchendiversifikation mit Neuentwicklung von grundlegenden Business-Objekten, Prozessen und Funktionen ist einer Neukonstruktion gleichzusetzen. Diese Erfahrungen für ein Expertensystem nutzbar zu machen, ist sicher zu aufwändig.

Anders sieht das bei der Variantenkonstruktion für kundenindividuelle Konfigurationen und für Branchenausprägungen durch Partner aus. Für diese Konstruktionsaufgaben ist eine Expertensystemunterstützung notwendig. Bei der Anwendung wissensbasierter Konfigurationssysteme für die Variantenkonstruktion gibt es Schwierigkeiten in der Erfüllung der beiden Voraussetzungen **Konfigurationsziele** und **Kontrollwissen**. Die anderen Voraussetzungen sind durch das Ordnungssystem für Anwendungselemente bereits erfüllt.

Es ist für ein betriebswirtschaftliches Anwendungssystem nicht möglich, alle Spezifikationen im Vorfeld festzulegen. Expertensysteme für die Konfiguration gehen häufig davon aus, dass im Wesentlichen die Anforderungen im Vorfeld definiert sein können. Lösungen mit interaktivem Dialog des Benutzers für die fortlaufende Spezifikation der Anforderungen gibt es zwar, sie werden jedoch außer mit Hinweisen auf die Nachteile des nicht-monotonen Schließens [Puppe91, 58] nicht ausreichend untersucht. Darum geht es aber bei der Anwendungssystementwicklung.

Die ***Methode der semantischen Komposition*** unterscheidet in der Variantenkonstruktion streng nach „Ermittlung der Teilaufgaben“ und „Auswahl und Komposition der Lösungselemente“.

Wissensbasierte Konfigurationssysteme streben es an, mit ihrem Kontrollwissen den gesamten Konstruktionsvorgang zu unterstützen. Nach der Spezifikation der Anforderungen erzeugt das System selbständig eine annehmbare Lösung. Erkannte Konflikte müssen im Verlauf des Konfigurationsvorgangs – durch Benutzereingriffe – aufgelöst werden. Ziel der „Expertensystemkonstruktion“ ist es jedoch, je nach Anspruchshaltung und Anwendungsbereich, eine qualitativ „ausreichende“ Konstruktionslösung zu erzeugen.

Die ***Methode der semantischen Komposition*** stützt sich für die Ermittlung der geeigneten Teilaufgaben auf die Ergebnisse der Unternehmensgestaltung, der Organisationsentwicklung sowie der Aufgaben- und Ablaufplanung ab. Es ist gerade eine Stärke dieses Ansatzes, dass die Ergebnisse unterschiedlicher Sichten und verschiedener Modellierungsmethoden in Form von Teilaufgaben für die flexible Komposition von Anwendungselementen eingesetzt werden können. Es ist zudem ein Bestreben der ***Methode der semantischen Komposition,*** durch möglichst orthogonale Anwendungselemente eine hohe Variabilität der herzustellenden Anwendungssysteme zu gewährleisten. Das Kontrollwissen muss angepasst an die Bedürfnisse der Anwenderunternehmen eine Vorgehensweise bei der Systemeinführung (Konfiguration) gewährleisten, die eine vom Kunden gewünschte Flexibilität nicht unnötig einschränkt. Dabei bewegt sich Konfiguration auf einer Gratwanderung zwischen betriebswirtschaftlichen Rahmenbedingungen, Anforderungen der Kunden aufgrund mangelnden Sachverstandes und fehlender Variabilität des Anwendungselemente-Baukastensystems.

Durch Referenz-Baugruppen stellt die ***Methode der semantischen Komposition*** Konstruktionshilfen zur Verfügung, welche die Freiheit der Komposition nicht einschränken.

Trotz aller vorbereitenden Maßnahmen und Referenzinformationen werden in jedem Einführungsprojekt bei Kunden die individuellen Anforderungen eines Unternehmens auf den Konstruktions- und Konfigurationsprozess Einfluss nehmen. Genau dann treten Konflikte bei Auswahl und Komposition von Anwendungselementen auf, für deren Lösung idealtypisch ein Expertensystem eingesetzt werden sollte. Die drei wesentlichen Konfliktlösungsstrategien im Bereich KI für Konfigurationsprobleme werden kurz dargestellt und anschließend im Kontext der ***Methode der semantischen Komposition*** diskutiert.

- **Ressourcenorientiertes Konfigurieren:** Hierbei werden, wie bei der Idee der „Module Interconnection Languages“, angebotene und angeforderte Ressourcen betrachtet. Sind alle angeforderten Ressourcen vorhanden, so existiert eine „gültige“ Konfiguration. Ressourcen können für eine exklusive oder eine gemeinsame Nutzung für andere Komponenten zur Verfügung stehen. Eine Prüfung auf Erfüllung aller Anforderungen erfolgt mittels eines Bilanzierungsverfahrens. Anforderungen an die Konfiguration selbst werden durch Ressourcenanforderungen der Systemumgebung dargestellt [Heinrich93, Gülden/Günter95]. Heinrich hat auch gleich Erweiterungen dieses „einfachen“ Modells angedeutet, womit die Bündelung von Ressourcen und die Vermittlung von Ressourcen über dritte Komponenten realisiert werden kann (siehe auch [Börding/Rahmer96]). Zudem gibt es für die Bilanzierung der Ressourcen effizienzsteigernde Verfahren, wobei durch Akkumulation von Bereitstellungen und Anforderungen Vergleichsprüfungen durchgeführt werden können. Dieses Verfahren ist immer NP-vollständig.

- **Strukturorientiertes Konfigurieren:** Beim strukturorientierten Konfigurieren werden die taxonomischen (abstraktiven) und kompositionalen Beziehungen zwischen Domänenobjekten für die Konfigurationsaufgabe eingesetzt [Gülden/Günter95, Tank93].

- **Fallbasiertes Konfigurieren:** Hierbei wird die Erfahrung vergangener Konfigurationslösungen für das aktuelle Konfigurationsproblem verwendet. Der Auswahl relevanter Konfigurierungsfälle, von deren Erfahrung profitiert werden soll, kommt eine besondere Gewichtung zu. Dabei gilt die Annahme: „Ähnliche Aufgaben haben ähnliche Lösungen“ [Pfitzner93, 28]. Durch Ähnlichkeitsmaße werden relevante Fälle (Lösungen) ermittelt und für die Problemlösung eingesetzt. Ein sehr interessanter Aspekt dabei ist, dass eine Zerlegung des Falls bzw. seiner Lösung in „maximal konfliktfreie Aggregate“ bezüglich der aktuellen Aufgabe eine Übernahme von Teillösungen zulässt [Pfitzner93, 29].

Die **ressourcenorientierte Konfiguration** wird in der ***Methode der semantischen Komposition*** nicht auf Schnittstellenebene durchgeführt. Für das ressourcen-

orientierte Konfigurieren können die Elemente der Implikationsbeziehung verwendet werden. Bei der Implikation ist diese Problemlösungsstrategie sinnvoll, denn sie stellt eine „gerichtete“ Beziehung zwischen Anwendungselementen dar. Dadurch kann das implizierende Element als anfordernde Ressource betrachtet werden und das implizierte Element als angebotene Ressource. Angebotene Ressourcen sind in der ***Methode der semantischen Komposition*** nicht als exklusiv definiert. Bedingt durch die verschiedenen Implikationstypen und durch die unterschiedlichen Struktur-Elemente, die für diese Form der Relation verwendet werden können, ist es eine Aufgabe des Konfigurationssystems, alle Anwendungselemente die für angebotene Ressourcen in Frage kommen zu ermitteln. Falls in der ***Methode der semantischen Komposition*** keine Einschränkungen bezüglich der Verwendung und Prüfung von Implikationsbeziehungen vorgenommen werden, so ist auch dieses Verfahren NP-vollständig. Eine Beherrschung der Abhängigkeiten wird durch die in Abschnitt 3.4.4 vorgeschlagenen fünf Regeln für das Beziehungsdesign der semantischen Relationen erwartet. Die Verpflichtung zur Einhaltung dieser Regeln verringert zwar die Ausdrucksstärke des Wissensrepräsentationsformalismus, kann jedoch die Antwortzeiten während der Konfiguration auf eine akzeptable Größe reduzieren. Für das Bündelungsproblem von Ressourcen können bei Bedarf mehrstellige Relationen eingeführt werden.

Strukturorientiertes Konfigurieren bedeutet für die ***Methode der semantischen Komposition*** eine Problemlösungsstrategie, welche die hierarchischen Beziehungen der Gliederung und der Baugruppenbildung verwendet, um eine optimale Lösungsstruktur zu ermitteln. Die Vorgehensweise kann man sich als Auflösung von Strukturstücklisten vorstellen [Tank93, 8].

Auch für das **fallbasierte Konfigurieren** hat die ***Methode der semantischen Komposition*** schon entsprechende Konstruktionshandlungen vorgesehen. Die Verwendung von Referenz-Baugruppen und von nicht standardisierten Projektergebnissen, die nur im Repository und nicht im Ordnungssystem gespeichert sind, entspricht dem Nutzen von Fallwissen. Pfitzner erläutert mehrere Möglichkeiten, ähnliche Fälle zu finden [Pfitzner93, 28ff]. Beispielsweise können anhand identischer oder ähnlicher Teilaufgaben geeignete Konstruktionsprojekte identifiziert werden. Nachdem ähnliche Fälle gefunden wurden, muss der Deckungsgrad mit der bestehenden Aufgabenstellung ermittelt werden. Eine gute Möglichkeit vergangene Konfigurationslösungen wiederzuverwenden ist dabei, Baugruppen in ihre maximal konfliktfreien Bestandteile zu zerlegen und diese als Ausgangspunkt für die neue

Konfiguration zu verwenden. Diese besondere Leistung ist Aufgabe eines Expertensystems.

Die ***Methode der semantischen Komposition*** sieht den Einsatz verschiedener Problemlösungsstrategien vor. Abhängig von den Konstruktionshandlungen und dem vorhandenen Wissen, können die Strategien für eine Konfigurationsaufgabe kombiniert werden. Diese Vorgehensweise wird auch in KONWERK, einem modularen domänenunabhängigen Konfigurierungswerkzeug (Ergebnis des PROKON-Projektes), favorisiert [Günter95c]. Nach Günters Meinung kann keine Anwendung im Bereich innovativer Konfiguration mit nur einer Problemlösungsstrategie auskommen.

Konfliktauflösung

Konflikte entstehen bei einer Konfiguration von Anwendungen aus Anwendungselementen durch eine Verletzung struktureller und/oder semantischer Beziehungen des Ordnungssystems. Einige Konflikte lassen sich lokal durch den „Methodenaufruf-Gültigkeitsbereich“ begrenzen. Dabei werden für bestimmte Abhängigkeiten (Alternativ-Komposition, Alternativ-Implikation, Negation) nur Prüfungen innerhalb des definierten Bereiches (z. B. benutzerspezifischer Arbeitsplatz) durchgeführt.

Werden Konflikte entdeckt, so gibt es verschiedene Möglichkeiten diese aufzulösen. Entweder durch Rücknahme von Konfigurierungsschritten (Backtracking) oder durch die Reparatur von Teilkonfigurationen [Günter93]. Für komplexe Konfigurationsaufgaben, die umfangreiche Neuberechungen bei Revisionsentscheidungen bedingen, werden eigenständige Abhängigkeitsverwaltungssysteme eingesetzt [Schädler95]. Verfahren zur Konfliktauflösung sollten zum einen den Berechungsaufwand für Konfigurationsänderungen minimieren, zum anderen müssen sie dem Benutzer sinnvolle Alternativen für die Konfliktauflösung vorschlagen.

Resumee

Für einen professionellen Einsatz der ***Methode der semantischen Komposition*** wird für die Konflikterkennung bei der Erfassung von Abhängigkeitsbeziehungen im Ordnungssystem und für die Konfliktvermeidung und Konfliktauflösung bei der Variantenkonstruktion die Integration eines Expertensystems empfohlen. Die

Erfahrungen der wissensbasierten Konfiguration dafür zu nutzen wäre sicher eine interessante Herausforderung.

3.4.6 Terminologiebasierter Aufbau des Ordnungssystems

Wir wollen in unserem Wissen vom Gebrauch

der Sprache eine Ordnung herstellen:

eine Ordnung zu einem bestimmten Zweck;

eine Ordnung von vielen möglichen Ordnungen;

nicht die Ordnung.

LUDWIG WITTGENSTEIN[43]

Aufbauend auf den in Abschnitt 2.3 eingeführten Grundlagen der sprachbasierten Anwendungsentwicklung werden in den folgenden Ausführungen die Anwendung und Erweiterung dieser Erkenntnisse hinsichtlich der ***Methode der semantischen Komposition*** dargestellt.

Die Verwendung einer Normsprache wird hier vorausgesetzt. Wenn also von Lexikon gesprochen wird, so ist darunter immer ein Normlexikon zu verstehen. Dasselbe gilt für Aussagen und Grammatik. Der Thesaurus des Lexikons wird auf der fachkonzeptionellen Ebene als Ordnungssystem für Termini verstanden und auch so bezeichnet.

Lexikon mit Ordnungssystem für Termini

Eine Normsprache ist nicht nur eine eindeutige, sondern auch eine reduzierte Sprache. Die offensichtlichen Nachteile der Reduktion können durch eine entsprechende Transparenz – was, wie und warum reduziert wurde – wieder ausgeglichen werden.

Für ein Lexikon mit Ordnungssystem bedeutet dies, dass sich der Vorgang der Normierung von Termini nicht nur durch das Zuordnen von Worterklärungen und Wortverwendungsbeispielen, sondern auch im Darstellen von Beziehungen zu anderen

[43] Zitiert nach [Wedekind81, 19]

Termini auf derselben Abstraktionsebene widerspiegelt. Es werden in diesem Ordnungssystem Synonymitätsbeziehungen („wird verwendet für“) und auch ausdrücklich untersagte Polysemiebeziehungen („bedeutet nicht“) aufgeführt[44]. Diese Beziehungen ermöglichen die Anpassung einer vorherrschenden Unternehmensfachsprache an die Sprache des Standard-Anwendungssystems. Dafür werden „Begriffe“ (eigentlich Bezeichner) des Anwenderunternehmens als individuelle Ergänzungen in das produktbegleitende Lexikon[45] des Herstellers aufgenommen und mit Synonymie- oder Nicht-Polysemiebeziehungen versehen. Werden neue Updates des Hersteller-Lexikons an Kunden ausgeliefert, so müssen die Individualergänzungen erhalten bleiben. Dasselbe gilt im Übrigen für anwenderindividuelle Ergänzungen von Begriffsdefinitionen.

Eine andere Möglichkeit die Standard-Anwendungssystem-Sprache an die Unternehmensfachsprache eines Anwenderunternehmens anzupassen besteht darin, die Anwendungselemente für ihren Gebrauch umzubenennen. Werden alle Verwendungsmöglichkeiten von Begriffsbezeichnungen in einem Anwendungssystem von der gleichen Textbasis (Lexikon) abgeleitet, d.h. Oberflächentexte, Texte in Fehlermeldungen, Texte in Formularen und Berichten sowie Texte in der Dokumentation, dann ist es möglich die Begriffsbezeichner kundenspezifisch oder branchenspezifisch zu ersetzten [Hüber et.al. 03]. Hüber u.a. gehen davon aus, dass eine kundenspezifische Anpassung nur für Großunternehmen wirtschaftlich sinnvoll ist, denn zum einen gewöhnen sich die Nutzer von Anwendungssystemen sehr schnell an die Begriffe und deren Bezeichner in „ihrem“ neuen System, sofern ihnen die Bedeutung vermittelt werden kann. Zum anderen ist dauerhaft ein Graben zwischen der Außenkommunikation des Herstellers (z.B. neue Produktinformation, Wartung und Support, Übungen in allgemeinen Trainingssystemen) und der Innenkommunikation des Anwenderunternehmens zu überbrücken. Völlig anders stellt sich die Situation für branchenspezifische Fachbegriffe und deren Bezeichner dar. Hüber u.a. nennen dies industriespezifische Ersetzung von Fachbegriffen (eigentlich Bezeichner). Damit ist die Anpassung der Anwendungssystem-Sprache an die Sprache einer Gruppe von

[44] Für den Fall, dass ein Polysem („sprachlicher Ausdruck mit mehreren Bedeutungen“ [Duden04]) als Deskriptor für eine seiner Bedeutungen verwendet wird, muss zu den Benennungen aller anderen ursprünglich umfassten Bedeutungen eine „bedeutet nicht“ – Beziehung hergestellt werden.

[45] Das Lexikon mit Ordnungssystem wird hier als wesentlicher Bestandteil der Produktdokumentation verstanden.

Unternehmen gemeint. Die Kosten für eine einmalige Umstellung (Ersetzung) der Fachbegriffe fällt dabei nur noch geringfügig ins Gewicht. Jedoch auch die Anpassung der Außenkommunikation des Herstellers ist wirtschaftlich vertretbar, da die Kosten dem Erschließen eines neuen Marktes (Branche) gegenüber stehen. Zudem gehen Hüber u.a. davon aus, dass Partner des Herstellers diese Aufgabe vollständig übernehmen können.

In Abschnitt 2.3 wurde schon erwähnt, dass Lexikonbegriffe einheitlich durch alle Entwicklungsphasen und für alle Abstraktionsebenen in der Anwendungssystem-entwicklung verwendet werden sollen. Daraus folgt, dass für die unterschiedlichen Ebenen der Konkretion „Fachentwurf“ (Aussagensammlung), „Systementwurf“ (Spezifikation), „Implementierung“ (Programmierung) auch verschieden detaillierte Informationen zur Worterklärung und Wortverwendung im Lexikon abgebildet sein müssen. Hier wird empfohlen, gerade für die Distribution des Lexikons an Anwender-unternehmen, zumindest eine Unterteilung der Begriffsdefinitionen in Anwendungs-bereichsdefinitionen (Fachentwurf) und Entwicklungsbereichsdefinitionen (System-entwurf und Implementierung) durchzuführen. Anhand eines Beispiels soll die Unterteilung der Terminusdefinitionen und die Relevanz der Verwendung normierter Termini in Fachentwurf, Systementwurf und Implementierung erläutert werden. In der *Kundenbestellung für Lagerartikel* gibt es einen *Auftragsgesamtpreis*. Diese Informationseinheit wird als Begriff im Lexikon hinterlegt.

Lexikoneintrag
Terminus: AUFTRAGSGESAMTPREIS
Fachentwurf: Auftragsgesamtpreis ist die Summe aller Positionspreise eines Auftrags. Gegebenenfalls kann diese Summe durch Rabattkonditionen des Auftrags, des Kunden oder der Positionsartikel korrigiert werden. Eine weitere Korrektur kann durch Versandkosten entstehen.
Systementwurf: Auftragsgesamtpreis = $\sum$ (Postitionspreis – Positionsrabatt) – (Kundenrabatt $\vee,\wedge$ Auftragsrabatt) + Versandkosten. Rabatte und Versandkosten sind optional und alternativ.
Wortverwendungsbeispiele: Auftragsgesamtpreis kann in Bestellaufträgen von Kunden und an Lieferanten verwendet werden. Auftragsgesamtpreis kann zur Berechnung der Versandkosten verwendet werden.

Tabelle 13: Beispiel eines Lexikoneintrags

In diesem Beispiel soll es eine Komponente geben, die für die Berechnung des Auftragsgesamtpreises in verschiedenen Anwendungselementen, die jeweils diese Informationseinheit enthalten, verantwortlich ist.

Name der Komponente: „Auftragsgesamtpreis“

Diese Komponente besitzt eine **Schnittstelle** IAuftragsgesamtPreis mit der **Methode** auftragsgesamtpreisBerechnen.

Aufgabe dieser Komponente ist es, die Einzelparameter wie Bestellpositionspreis, Rabattkonditionen, Versandkosten etc. zu ermitteln und entsprechend eines Berechnungsschemas daraus den Auftragsgesamtpreis zu berechnen. Diese Funktion ist ebenfalls Bestandteil der Komponente und liefert als Ergebnis ihrer Berechnung den Auftragsgesamtpreis in den Rückgabe-Parameter (auftragsgesamtpreis:double) der Methode auftragsgesamtpreisBerechnen.

Dieses kleine Beispiel verdeutlicht, wie wichtig eine durchgängige Verwendung von normierten Termini für eine Anwendungsentwicklung ist. An vielen Stellen in einem System werden fachlich determinierte Begriffe in der Programmierung verwendet. Wenn nicht klar ersichtlich ist, was diese Begriffe bedeuten, so sind auch die Inhalte der Programmkonstrukte nicht zu verstehen und folglich auch nicht zu prüfen. Weder Schnittstellen noch Komponenten oder Funktionen können auf ihre korrektes Design und ihre korrekte Implementierung geprüft werden, wenn ihre Benennung nicht mehr im Zusammenhang mit ihrem fachlichen Inhalt steht. Wenn beispielsweise die Komponente mit „Gesamtpreis“ bezeichnet würde oder die Funktion mit „Preisberechnung“ usw.

Dasselbe gilt für alle Konstrukte der Anwendungsentwicklung, die mit einem Bezeichner versehen werden können (z.B. Klassen, Attribute, Operationen, Tabellen, Abhängigkeitsbeziehungen, etc.). Es gilt ebenso für alle Elemente des Ordnungssystems für Anwendungselemente, denn nur dann kann die Verständnisebene (Abstraktionsebene) der Anwender für die Beschreibung von Anwendungselementen getroffen werden, wenn die Bezeichner der Sachmerkmalleisten, Merkmale und Ausprägungen sowie aller Gliederungsbegriffe der normierten Fachsprache des Anwendungsbereiches entstammen. Wird der Aufbau eines Ordnungssystems in dieser Weise auf die (normierte) Terminologie eines Anwendungsbereiches abgestützt, kann von einem terminologiebasierten Ordnungssystem gesprochen werden.

In diesem Abschnitt wurden bisher zwei verschiedene Ordnungssysteme angesprochen:

1. Ordnungssystem für das Lexikon als Teil der Terminologie des Anwendungsbereichs
2. Ordnungssystem für Anwendungselemente

Das Ordnungssystem des Lexikons und das Ordnungssystem für Anwendungselemente stehen in einer definierten Beziehung. Zur Erläuterung dieser Inter-Ordnungssystem-Beziehung wird in folgender Grafik ein Ausschnitt eines Beziehungsgraphen für Thesaurusbeziehungen eines Lexikons dargestellt.

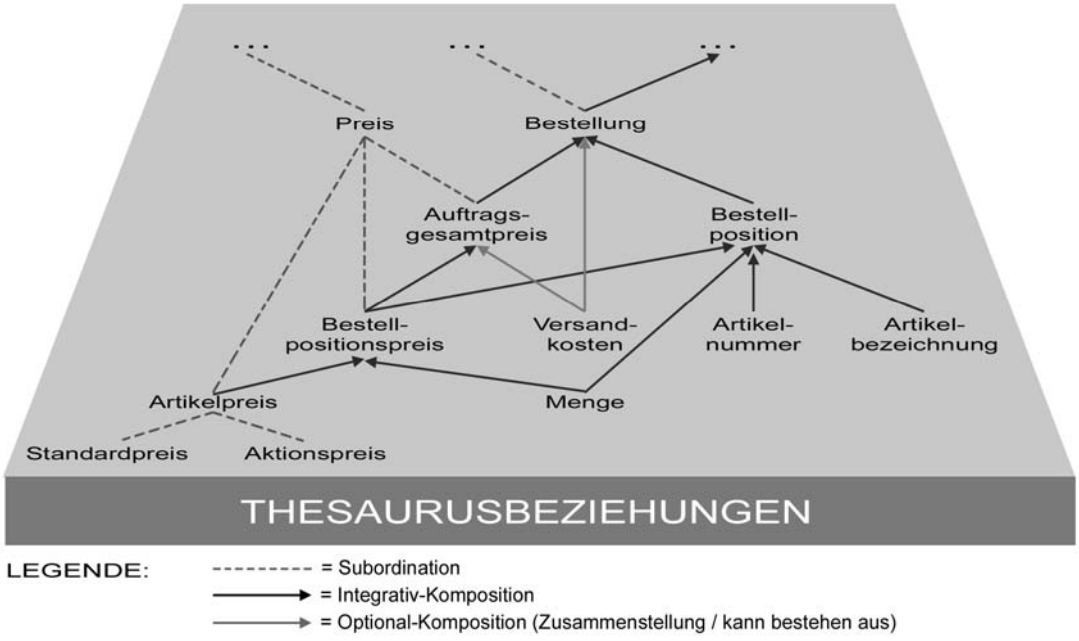


Abbildung 34: Ausschnitt eines Beziehungsgraphen für Thesaurusbeziehungen

Die hier aufgeführten Thesaurusbeziehungen zeigen nicht das gesamte Spektrum an möglichen Beziehungen (siehe Abschnitt 2.3). Zum besseren Vergleich wurden sie mit denselben Bezeichnungen wie die strukturellen Beziehungen der Gliederung benannt. Inhaltlich entstammt dieses Beispiel zwar der Herstellungsebene, es kann jedoch sehr anschaulich die Problematik verdeutlichen, die eine Unterscheidung zwischen Thesaurus (Aufbau der Sprache) und Gliederung (Aufbau des Anwendungssystems bzw. Anwendungselemente-Baukastensystems) notwendig macht.

Das Lexikon mit seinem Ordnungssystem stellt die Terminologie, d.h. die Sprache eines Anwendungsbereiches dar. Im Betrieb befindliche Anwendungssysteme sind folglich Sprachwerke, wie auch Bücher eine Anwendung der Umgangssprache darstellen [Ortner97, 82ff.].

Ein Baukastensystem für Anwendungselemente mit seinem Ordnungssystem enthält viele Sprachwerk-Bauteile (Anwendungselemente, Struktur-Elemente, strukturelle und semantische Beziehungen). Für diese Sprachwerk-Bauteile gilt, wie für Sprachwerke auch, sie können nicht alle in der Sprache enthalten bzw. vorgedacht sein.

Alle Begriffe der Sprachwerke müssen im Lexikon enthalten sein. Für die „Wortverwendung“ als Beschreibungselement des Lexikons ist es ersichtlich, dass nicht alle Varianten einer Wortverwendung in einem Lexikon enthalten sein können. Für Thesaurusbeziehungen ist die Entscheidungsfrage: Welche Begriffsbeziehungen eines Anwendungssystems gehören noch in den Thesaurus und welche nicht mehr? nicht immer eindeutig zu beantworten.

Das Beispiel in Abbildung 34 zeigt, wie eine Bestellung aus ihren Einzelteilen hergestellt werden kann. Der optionale Aspekt, der in der Grafik durch die Optional-Komposition zum Ausdruck gebracht wird, existiert in der Klassifikations- und Thesaurusforschung der DGD (Deutsche Gesellschaft für Dokumentation) nicht [Schmitz-Esser95]. Er könnte aber als eigene Beziehungsart definiert werden. Die Ausdrucksstärke der Begriffsrepräsentation ist schwächer, wenn optionale Beziehungen im Thesaurus als absolut oder überhaupt nicht aufgeführt werden. Das ist jedoch in diesem Zusammenhang nicht maßgebend. Wesentlich ist, dass in einem Thesaurus keine Designentscheidungen für den Aufbau eines Anwendungssystems vorweggenommen werden dürfen. Es gibt inhaltlich korrekte Beziehungen, die in einem Thesaurus abgebildet werden müssen. Sie können auch unmittelbar auf die Entwicklung eines Baukastensystems einwirken. Allerdings müssen diese Beziehungen von allen Fachleuten für inhaltlich notwendig erklärt werden.

Es kann nicht die ganze Vielfalt an Kompositionsmöglichkeiten des Baukastensystems (alle Referenzmodelle) in einem Thesaurus abgebildet werden und auf keinen Fall individuelle Zusammenstellungen von Anwendungselementen eines bestimmten Anwenderunternehmens. Der Thesaurus wird vom Hersteller des Baukastensystems erstellt und gewartet und an alle Anwenderunternehmen ausgeliefert. Abbildungen von

Referenz-Baugruppen dürfen in einem Thesaurus allenfalls exemplarisch enthalten sein.

Subsumtionsbeziehungen zwischen Anwendungselementen und Sachmerkmalleisten werden nicht im Thesaurus repräsentiert, da ein Thesaurus nur die Beziehungen zwischen Begriffen und nicht zwischen Exemplaren abbildet. Die Subordinationsbeziehungen der Gliederung und des Thesaurus hingegen sollten identisch sein. Synonymitäts- und Polysemiebeziehungen des Thesaurus werden nicht in der Gliederung abgebildet. In der Gliederung befinden sich nur so genannte Deskriptoren, das sind Vorzugsbenennungen von Begriffen. Außerhalb der Gliederung gibt es noch die Eigenschaften der Anwendungselemente in Form von Merkmalen und Ausprägungen. Diese Begriffe müssen als Deskriptoren im Lexikon vorhanden sein. Ihre Beziehungen zu Sachmerkmalleisten werden nicht im Thesaurus aufgeführt.

Eine umfassende Regel gilt als unverletzlich:

> Die Verwendung einer Sprache darf ihrer Definition (Begriffsdefinition und definierten Begriffsbeziehungen) nicht widersprechen.

Falls Sprachwerke eine nicht zulässige Verwendung der Sprache fordern, so muss eine Rückkopplung für die Korrektur des Lexikons oder von Thesaurusbeziehungen stattfinden.

Komponentenorientierter Systementwurf aus normsprachlichen Aussagen

Aufbauend auf dem methodenneutralen Fachentwurf von Ortner [Ortner97] wird hier der Übergang zum komponentenorientierten Systementwurf erörtert. Das Ergebnis des methodenneutralen Fachentwurfs ist eine Sammlung terminologisch und grammatisch normierter Aussagen, die alle Aspekte eines Anwendungssystems beschreiben können. Diese Aussagen werden entsprechend den eingesetzten Lösungsparadigmen weiterverarbeitet.

Für die Entwicklung von Anwendungssystemen können unterschiedliche Lösungsparadigmen konkurrierend oder ergänzend eingesetzt werden (z.B. „objektorientierte Systementwicklung“, „datenbankorientierte Lösungen“, „Workflow-Anwendungen“ oder „komponentenorientierte Anwendungen“).

Jedes Lösungsparadigma besitzt (eine) eigene Spezifikationssprache(n). Eine Aussagensammlung des methodenneutralen Fachentwurfs sollte in alle Spezifikationssprachen transformierbar sein. Für den objektorientierten Systementwurf und den Systementwurf von Workflow-Management-Anwendungen wurden von Schienmann [Schienmann97] und Lehmann [Lehmann99] die Möglichkeiten der Transformation nachgewiesen[46].

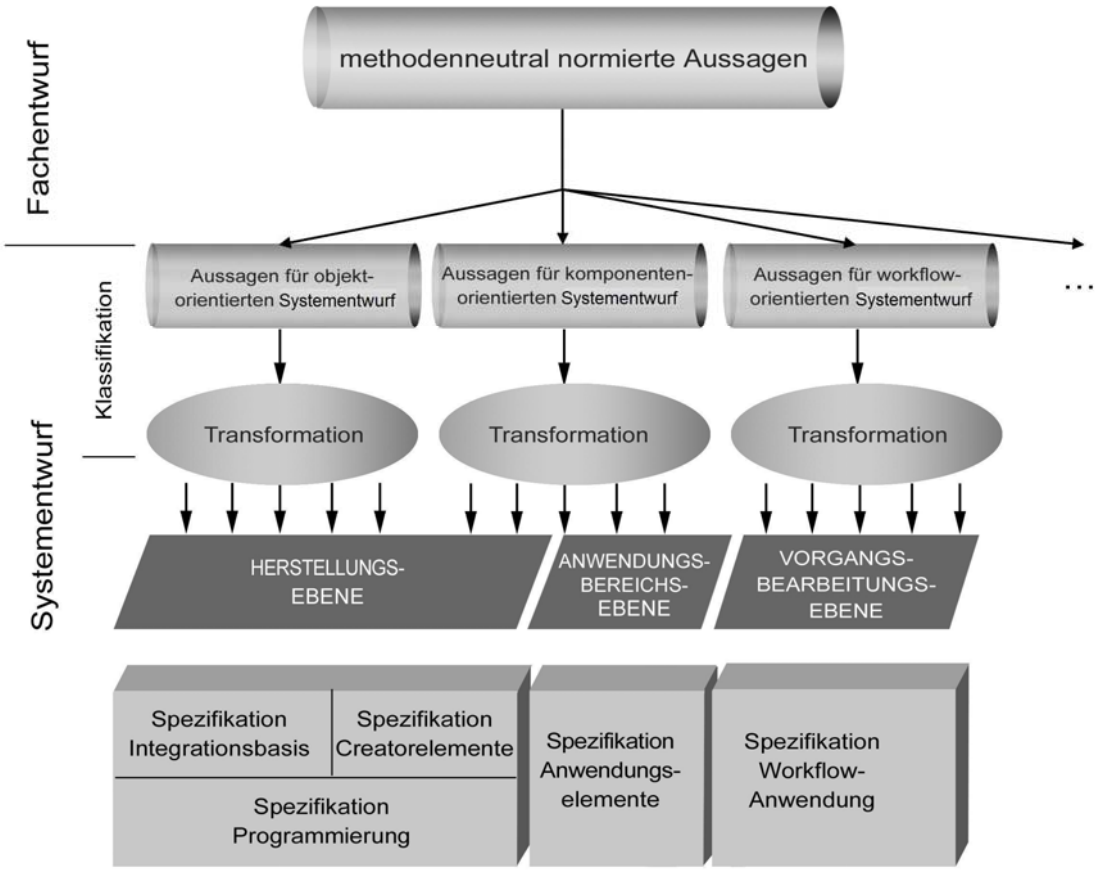


Abbildung 35: Aussagentransformation

Alle Systementwicklungsmethoden haben eines gemeinsam, der Übergang von einer Beschreibungsebene zur nächsten lässt immer Gestaltungsspielraum für die Transformation der einzelnen Methodenkonstrukte zu (Methodenneutraler Fachentwurf ➔ Systementwurf ➔ Implementierung). Zwischen den Beschreibungsebenen findet

[46] Die Idee eines methodenspezifischen Fachentwurfs wurde zu Gunsten einer verbesserten Zuordnung der methodenspezifischen Aussagen zum Systementwurf aufgegeben.

sinnvollerweise immer eine Rückkopplung statt. Letztlich können nur Entscheidungen für bestimmte Implementierungstechnologien und daraus abgeleitete Entwurfs- und Gestaltungsanweisungen die „bottom-up“ entwickelt wurden stringente Transformationen ermöglichen.

Bevor eine Transformation stattfinden kann, muss eine Klassifikation und Modifikation der methodenneutralen Aussagen für ihre methodenspezifische Verwendung durchgeführt werden. Dazu werden die Aussagetypen (Aussageformate, Satzbaupläne) des methodenneutralen Entwurfs den Repräsentationskonstrukten der Ziel-Spezifikationssprache zugeordnet. Für den Transformationsvorgang müssen eindeutige Verknüpfungen zwischen Aussagetypen und Repräsentationskonstrukten hergestellt werden können.

Eine Transformation kann folgendermaßen aussehen:

Aussage → Aussagentyp ➔ Repräsentationskonstrukt → inhaltliche Spezifikation

Beispiel:
„Kundenbestellung **hat** Rahmenvereinbarung“ → *X **hat** Y* ➔ *Anwendungselemente **haben** Merkmale* → Anwendungselement: *Kundenbestellung* hat das Merkmal: *Rahmenvereinbarung*

Dabei können folgende Schwierigkeiten auftreten [vgl. Schienmann97, 172]:

1. Unter Umständen können erst mehrere ergänzende Aussagetypen ein Repräsentationskonstrukt der Spezifikationssprache vollständig beschreiben.
2. Mehrere Aussagen beschreiben denselben Aspekt der Anwendung.
3. Eine normierte Aussage respektive ihr Aussagentyp findet keine Entsprechung in der Spezifikationssprache.
4. Für einen Aussagentyp gibt es mehrere zulässige Repräsentationskonstrukte der Spezifikationssprache mit unterschiedlichen Bedeutungen. Sie stellen den Handlungsspielraum der Analyse dar.

Für einen komponentenorientierten Systementwurf wird folgende Behandlung der Aussagen vorgeschlagen[47]:

1. Normierten Aussagen werden für die Beschreibung eines exemplarischen Repräsentationskonstruktes in der ***Methode der semantischen Komposition -*** soweit möglich - auf eine Aussage reduziert (Konzentration).

2. Anschließend werden die Aussagen für die Abbildung branchenweiter Standardlösungen systematisch variiert (Variation).

3. Existieren Aussagen, die nicht eindeutig einem Repräsentationskonstrukt der ***Methode der semantischen Komposition*** zuzuordnen sind, die jedoch auf jeden Fall diesem Lösungsparadigma angehören, so werden sie durch die Entwurfsentscheidungen eines Systemanalytikers hin zu einer geeigneten Aussage modifiziert.

Nach der ***Methode der semantischen Komposition*** erstellte Aussagensammlungen sind relativ einfach strukturiert, da sich die Modellierung dieser Methode auf die Variabilitäten eines Anwendungssystems, d.h. auf die anwenderindividuelle Gestaltung der Anwendung konzentriert.

[47] Jede Veränderung von Aussagen muss nachvollziehbar dokumentiert werden.

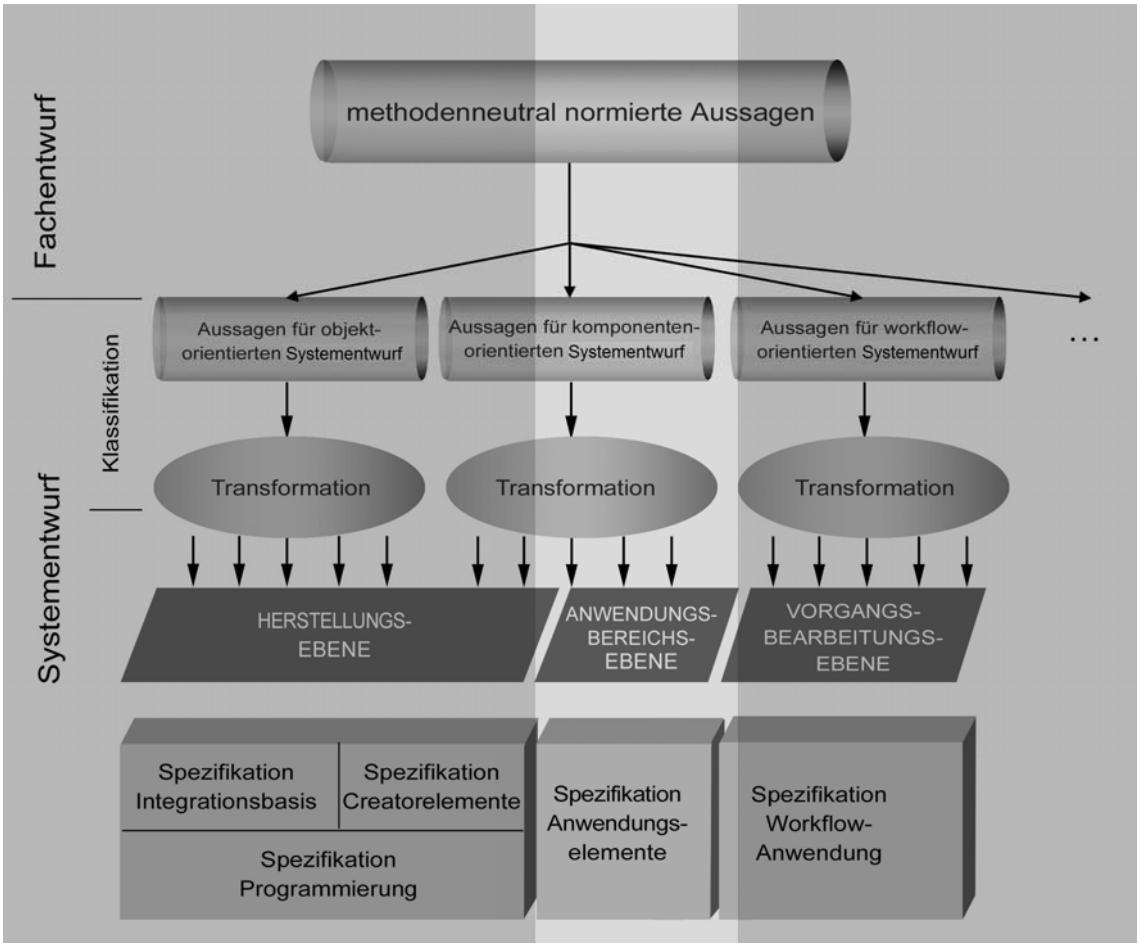


Abbildung 36: Aussagentransformation der Methode der semantischen Komposition

Die folgende Tabelle zeigt einen Ausschnitt aus möglichen Aussagetypen mit ihren Schlüsselworten. Aussagen wie „X ε (ist ein) Merkmal“ werden als Zuordnung von Begriffen zu Strukturelementen der ***Methode der semantischen Komposition*** vorausgesetzt.

Aussagetypen	Struktur-Elemente und Schlüsselworte	Beispiele
Subordination	{Unterbegriff}X ist ein {Oberbegriff}Y	Kundenbestellung ist eine Bestellung.
Subsumtion	{Merkmal}X gehört zu {Sachmerkmalleiste}Y	Rahmenvereinbarung gehört zu Kundenbestellung.
Komposition	{Baugruppe}X besteht aus {Bestandteil}Y	Zubehörverkauf besteht aus Kundenbestellung.
Anwendungselemente	{Anwendungselement}X hat {Eigenschaft}Y	*Kundenbestellung für Lagerartikel* hat Rahmenvereinbarung: Auftragsrabatt.

Tabelle 14: Aussagetypen mit Schlüsselworten

Für semantische Relationen wurden in Abschnitt 3.4.4 schon Beispiele für Satzstrukturen und Schlüsselworte aufgeführt. Die Kompositionsbeziehungen können wie die semantischen Relationen gleichfalls weiter zerlegt und entsprechend identifiziert werden.

Es lassen sich viele Aussageformen und Schlüsselworte bestimmen, die auf eine Repräsentation in der ***Methode der semantischen Komposition*** überführt werden können. Damit ist jedoch nicht das Problem vieler Entwurfsaufgaben gelöst. Gerade die oben vorausgesetzte Zuordnung von Begriffen zu den Struktur-Elementen der ***Methode der semantischen Komposition*** ist ein Problem, welches durch die Entscheidungsfreiheit der Modellierung auf verschiedenen Abstraktionsebenen noch verstärkt wird.

- Wann ist ein Begriff ein Anwendungselement?
- Wann ist ein Begriff eine Eigenschaft?
- Wann ist ein Begriff ein Bestandteil?
- Wann ist ein Begriff eine Sachmerkmalleiste?
- Wie wird ein Anwendungselement richtig begrenzt bzw. welche Eigenschaften gehören zu demselben Anwendungselement?

Durch die Konstruktionshandlungen der ***Methode der semantischen Komposition*** können diese Fragen zum Teil beantwortet werden. Anwendungselemente entsprechen den Aufgaben im Anwendungsbereich, also müssen Fragen wie: „Ist *Kundenbestellung editieren* eine Aufgabe aus dem Anwendungsbereich?“ gestellt werden. Anschließend muss geklärt werden, welche Eigenschaften ein Anwendungselement zur Bearbeitung der Aufgabe besitzen (haben) sollte. Eigenschaften werden als Ausprägungen erkannt. Ausprägungen werden zu Merkmalen zusammengefasst und diese einer Sachmerkmalleiste zugeordnet. Durch eine Kombination der Merkmalausprägungen einer Sachmerkmalleiste muss ein Anwendungselement beschrieben werden können. Eine systematische Variation dieses Anwendungselementes erweitert die Aufgabenbearbeitung im Anwendungsbereich.

Die Erstellung methodenspezifischer Aussagen geschieht in mehreren Reduktions- und Modifikationsschritten:

1. Reduktion: Normierung der Gebrauchssprache.
2. Reduktion: Konzentration von normierten Aussagen eines Sachverhaltes auf die „Dokumentation“ genau einer Spezifikationsaussage.
3. Modifikation von methodenspezifischen Aussagen, um bestimmte Designentscheidungen zu unterstützen (Rückkopplung zwischen Konstruktionsmethode und Aussagenbestand).
4. Variation der Aussagen, um die Variabilität des Anwendungssystems zu erhöhen.

Auch für die Verwaltung des Aussagenbestandes ist ein Ordnungssystem notwendig. Dieses Ordnungssystem interagiert ebenfalls mit dem Ordnungssystem für Anwendungselemente, denn es muss entsprechend dessen Struktur- und Beziehungselementen strukturiert werden. Eine permanente Angleichung und Rückkopplung ist notwendig. In nachfolgender Grafik wird die terminologische Basis der ***Methode der semantischen Komposition*** im Überblick dargestellt. Die Grundforderung, dass ausschließlich Termini des Lexikons für die Begriffe des Ordnungssystems für Anwendungselemente verwendet werden dürfen, wurde in der Grafik zugunsten der Übersichtlichkeit nicht eingezeichnet.

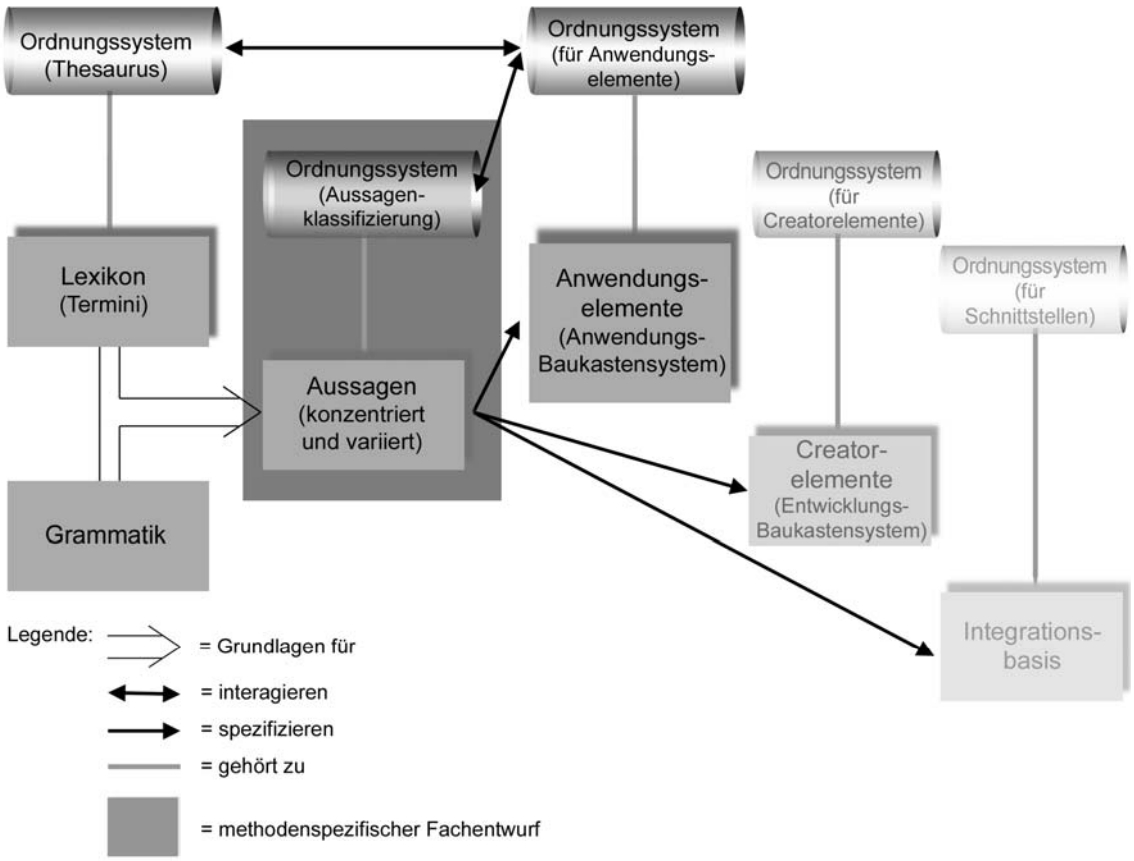


Abbildung 37: Terminologische Basis der Methode der semantischen Komposition

Mit steigender Konstruktionserfahrung der ***Methode der semantischen Komposition*** werden die Schlüsselworte und die Satzbaupläne der methodenspezifischen Aussagen immer griffiger und entwickeln sich dahin, dass eine sukzessive und reduktive Zuordnung der Aussagen von der Gebrauchssprache bis hin zum Ordnungssystem für Anwendungselemente möglich ist.

Eine vollständige Aufteilung des Aussagenbestandes in die einzelnen Lösungsparadigmen ist nicht immer möglich. Ein Ordnungssystem für Aussagen mit polyhierarchischen Zuordnungsstrukturen ermöglicht die Verwaltung von methodenspezifischen Aussagen unterschiedlicher Lösungsparadigmen in einem Aussagenbestand.

Der folgende 5-Schritte-Plan soll in Form einer Anleitung den Aufbau eines terminologiebasierten Ordnungssystems noch einmal zusammenfassen:

Schritt 1: Erstellung einer normierten Terminologie des Anwendungsbereiches.

Schritt 2: Aufbau eines Lexikons mit Thesaurus zur Verwaltung der normierten Fachbegriffe.

Schritt 3: Termini, die Aufgaben bezeichnen, werden als Anwendungselemente modelliert. Die Eigenschaften der Aufgaben werden als Ausprägungen erkannt, zu Merkmalen zusammengefasst und den Anwendungselementen zugeordnet. Verschiedene Merkmalausprägungen stehen für mögliche Varianten der Anwendungselemente. Klassen von Anwendungselementen werden zu Sachmerkmalleisten zusammengefasst. Sachmerkmalleisten werden durch systematische Variation erweitert.

Schritt 4: Das Wissen des Anwendungsbereiches wird abstraktiv und kompositiv polyhierarchisch in einer Gliederung hinterlegt.

Schritt 5: Untersuchung der Aussagen über die Komposition von Anwendungselementen und Sachmerkmalleisten (unterschiedlicher Abstraktionsebenen) hinsichtlich der Schlüsselwörter zur Identifikation von semantischen Relationen. Aufbau des semantischen Beziehungsnetzes.

Anwendungselemente unterliegen entsprechend den Begriffen einer Sprache ebenfalls den Widrigkeiten der Verschiebung von Begriffswelten [Hansen et al. 92]. Sowohl Begriffe und deren Bedeutungen als auch die inhaltlichen Beziehungen zwischen den Begriffen werden sich laufend verändern. Übertragen auf das hier vorgestellte Ordnungssystem für Anwendungselemente werden sich permanent neue Sachmerkmalleisten im Zeitverlauf, und bedingt durch die verschiedenen Abstraktionsbedürfnisse unterschiedlicher Anwendergruppen, auf übergeordneter Ebene bilden oder auf untergeordneter Ebene ausdifferenzieren. Weiterhin werden sich Verschiebungen der Termini zwischen Ausprägung, Merkmal und Sachmerkmalleiste ergeben. Zudem sind die Anwendungselemente an sich auch Veränderungen unterworfen. Feste Baugruppen (mit eigener Sachmerkmalleiste) werden neu konstruiert und hergestellt und aus Gründen mangelnder Orthogonalität werden elementare

Anwendungselemente von bestehenden Anwendungselementen abgespalten. Wechselwirkungen zwischen Anwendungselementen entstehen neu, werden umfangreicher, verändert oder aufgelöst.

Ein Werkzeug, welches diese „Anwendungselemente-Sprache“ als Grundlage für die Komposition von Anwendungen verwenden möchte, muss ihre Verwaltung in wechselseitiger Beeinflussung mit der Fachsprache des Anwendungsbereiches unterstützen und dies den Anwendern des Composer-Tools transparent machen. Die Schritte drei bis fünf des oben aufgeführten Plans müssen permanent von Neuem durchlaufen werden um diesen Anforderungen gerecht zu werden. Auch die Schritte eins und zwei sind nicht statisch. Sie sollten jedoch längerfristigen Bestand haben, denn eine Änderung der lexikalischen Bedeutung von Termini hat weitreichende Auswirkungen auf das gesamte System.

3.5 Zusammenfassung der Methode der semantischen Komposition

In der bisherigen Arbeit wurde die ***Methode der semantischen Komposition*** als eine Konstruktionsmethode für die Entwicklung, Variantenbildung und Kundenkonfiguration von betriebswirtschaftlichen Standard-Anwendungssystemen vorgestellt.

Für eine Verbesserung der Kommunikation zwischen Anwendern und Anwendungsbereichsexperten auf der einen Seite und Entwicklern auf der anderen Seite wurde eine Kommunikationsbasis bestehend aus **aufgabenorientierten Anwendungselementen** geschaffen.

Anwendungselemente repräsentieren die Variabilität eines Standard-Anwendungssystems. Der Einsatz von Anwendungselementen wird in Form eines **Baukastensystems** realisiert. Die Grundzüge von Anwendungselemente-Baukastensystemen wurden dargestellt, im Besonderen die Einflüsse von **Filtersystemen** auf das Anwendungselemente-Angebot in einem Baukastensystem.

Ausgewählte Konstruktionsprinzipien zum Aufbau und Ausbau eines Baukastensystems sind Bestandteil der ***Methode der semantischen Komposition***. Sie bieten im Speziellen Hilfestellung zur konstruktiven Ermittlung der geeigneten Granularität von Anwendungselementen mit dem Ziel einer höchstmöglichen Kombinierbarkeit. Das Verfahren der **Orthogonalisierung** wurde hierfür entwickelt.

Durch dessen konsequenten Einsatz können die elementaren Faktoren für den Aufbau von Anwendungssystemen erkannt werden.

Als Architektur-Beschreibungs-Sprache im Sinne von Shaw und Garlan [Shaw/Garlan96, 147ff] wurde ein **Ordnungssystem für Anwendungselemente** entworfen. Für das Ordnungssystem wurde nach dem Vorbild technischer Konstruktionsbereiche ein Katalog-System für die Verwaltung der Anwendungselemente entwickelt. Ein **Konstruktionskatalog** besteht aus Anwendungselementen (Hauptteil), **Sachmerkmalleisten** (Zugriffsmerkmale) und einer **polyhierarchischen Gliederung**. Die Verwaltung von **semantischen Relationen** für die Abbildung von Abhängigkeitsbeziehungen zwischen Anwendungselementen vervollständigt das Ordnungssystem.

Dieses Ordnungssystem wurde auf eine terminologische Basis gestellt. Alle Fachbegriffe des Ordnungssystems: Ausprägungen, Merkmale, Sachmerkmalleisten und Gliederungsbegriffe werden als Termini in einem Lexikon verwaltet. Alle Strukturelemente, strukturelle und semantische Beziehungen werden durch normsprachliche Aussagen begründet und dokumentiert. Diese starke Einbindung der Terminologie des Anwendungsbereiches gilt jedoch nicht nur für das Ordnungssystem und den fachkonzeptionellen Teil der Anwendungselemente, sondern auch für ihre Herstellung.

Das Ordnungssystem übernimmt verschiedene Aufgaben innerhalb der ***Methode der semantischen Komposition***. Es repräsentiert die Wissensorganisation des Anwendungsbereiches. Dadurch stellt es für die Neukonstruktion eines Baukastensystems die Entwicklungsvorgaben bereit. Für die **Variantenkonstruktion**, das „Konfigurations-Konzept“ der ***Methode der semantischen Komposition***, werden Branchenbaupläne (Referenzmodelle) bzw. konkrete Anwendungsbaupläne (für ein bestimmtes Unternehmen) durch Konstruktionshandlungen extrahiert. Entsprechend den Anwendungsbauplänen werden die Anwendungen aus den Bausteinen des Baukastensystems kundenindividuell konfiguriert.

Das Hauptziel der ***Methode der semantischen Komposition*** ist, die Variantenkonstruktion so erfolgreich wie möglich zu gestalten. Dazu wurden verschiedene Prinzipien und Lösungen entwickelt und eingesetzt:

1. Eine schrittweise Informationsversorgung: „Ordnungssystem – Anwendungselemente – zusätzliche Repository-Informationen“, reduzieren die zu verarbeitende Informationsmenge auf eine der Konstruktionshandlung entsprechende notwendige Größe.

2. Der Einsatz einer normierten Terminologie und der Zugriff dazu über ein Lexikon vereinheitlicht die Sprache der Konstrukteure (Composer und Creator) und die Sprache im Konstruktionskatalog.

3. Die technologieunabhängige Beschreibung der Anwendungselemente ermöglicht es, einem Anwendungsbereichsexperten (Composer) die geeigneten Anwendungselemente aus dem Katalog auszuwählen.

4. Nur durch das „reine Auswahlprinzip“ ist es einem Composer möglich, die Komposition selbstständig durchzuführen.

5. Ein gleichzeitiges Bewegen auf verschiedenen Abstraktionsebenen erfordert nur dort eine intensive Individualisierung und somit Auseinandersetzung mit Detailinformationen, wo ein Anwenderunternehmen dafür den Nutzen erkennt (Wettbewerbs-differenzierung).

6. Die Unabhängigkeit der ***Methode der semantischen Komposition*** von den Methoden zur Ermittlung der Teilaufgaben bedeutet für den Composer Freiheit bei der Methodenwahl für die Gestaltung der Geschäftsabläufe und –inhalte. Für die ***Methode der semantischen Komposition*** bedeutet es Neutralität gegenüber der Weiter-entwicklung von Methoden für die Geschäftsmodellierung.

7. Der Aufwand für das „Aussortieren“ von nichtgeeigneten Bausteinen aufgrund schon getroffener Entscheidungen wird durch den Einsatz von Filtersystemen reduziert.

8. Die Beherrschung der Komplexität inhaltlicher Abhängigkeiten zwischen Anwendungselementen wird durch die Verwaltung und Prüfung der semantischen Relationen sichergestellt.

Die semantischen Relationen sind ausdrücklich dafür geeignet, für die Konfigurations-unterstützung wissensbasierte Systeme einzusetzen. Dabei zeigt sich die Tragfähigkeit des Ordnungssystems für Anwendungselemente. Denn nur durch die klaren

Strukturen, die Möglichkeiten der Beziehungsverwaltung und die Einhaltung von eingeführten Regeln lässt sich der Berechnungsaufwand für eine konfliktfreie Konfiguration auf einen polynomialen Zeitbedarf reduzieren. Eine abschließende Grafik zeigt einen skizzierten Überblick der ***Methode der semantischen Komposition***.

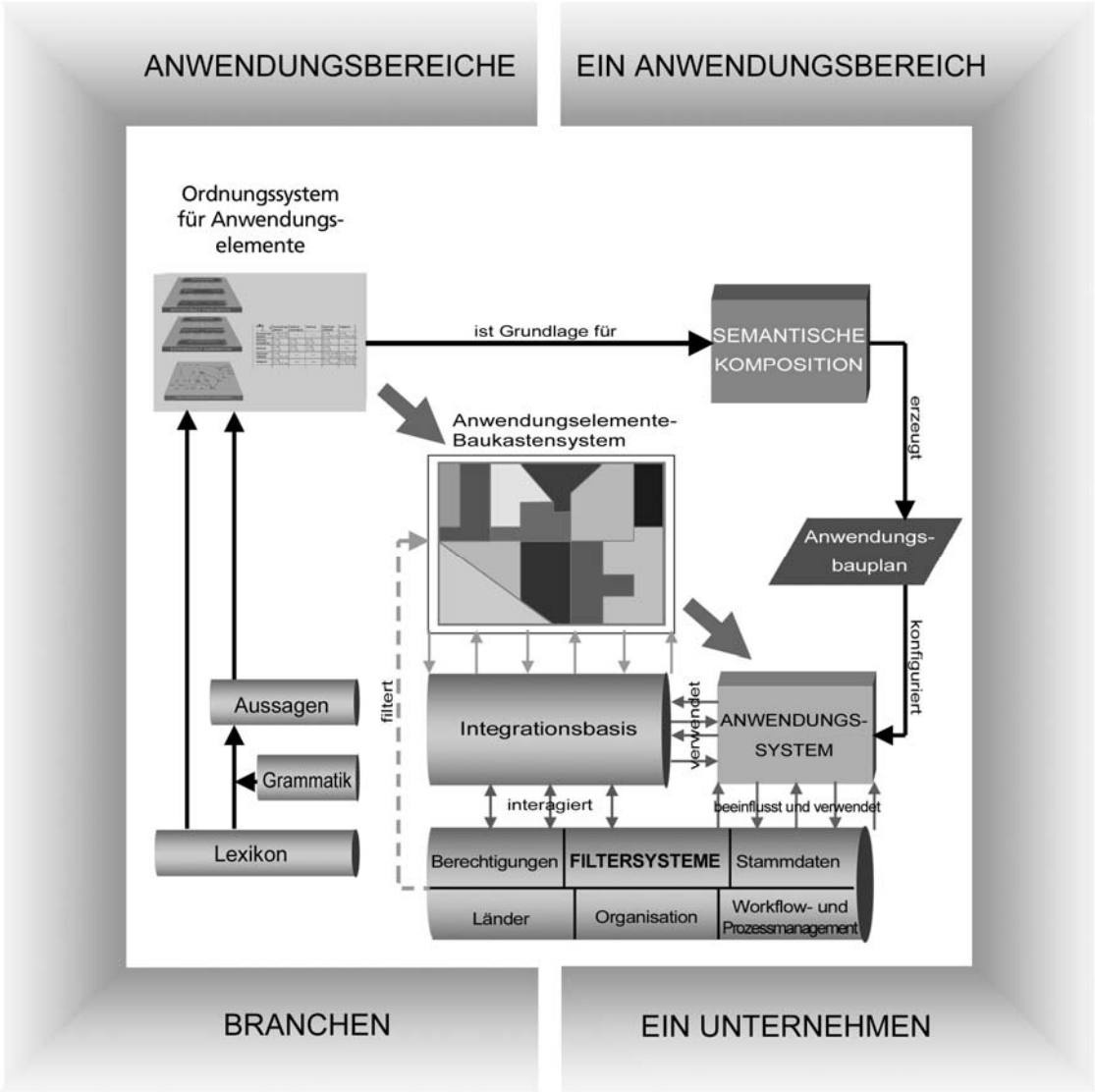


Abbildung 38: Übersicht der Methode der semantischen Komposition

Erst die Leistungsfähigkeit der ***Methode der semantischen Komposition*** eine unternehmensindividuelle Variantenkonstruktion zu unterstützen, ermöglicht den Anwenderunternehmen eine Selbstkontrolle ihrer Anwendungssysteme. Dies gilt sowohl für die Einführung als auch für die Weiterentwicklung durch Katalogbestellung neuer Anwendungselemente.

Kapitel 4 Anwendung der Methode der semantischen Komposition auf die Konstruktion von Standard-Anwendungssystemen

Für die Verifikation der vorgestellten Konstruktionsmethode wurde im Bereich Wirtschaftsinformatik I, Entwicklung von Anwendungssystemen, der Technischen Universität Darmstadt, ein funktionsfähiger Forschungsprototyp zur Verwaltung von Anwendungselementen und Creatorelementen, zum Aufbau eines Ordnungssystems und einer Fachterminologie sowie zur Variantenkonstruktion von Beispielanwendungen entwickelt. Mit dem Projektnamen „Terminologiebasiertes Komponenten-Management-

System“ wird das Ergebnis hier vorgestellt. In der Folge wird ein Anwendungsszenario für die Vorgehensweise der ***Methode der semantischen Komposition*** innerhalb des Multipfad-Vorgehensmodells (siehe Abschnitt 2.3) anhand eines Fallbeispiels dargestellt.

4.1 Terminologiebasiertes Komponenten-Management-System (TKMS)

Für das Projekt „Terminologiebasiertes Komponenten-Management-System“ (***TKMS***) wurde für Forschung und Lehre am Lehrstuhl von Prof. Dr. Erich Ortner, Fachgebiet Wirtschaftsinformatik I – Entwicklung von Anwendungssystemen am Institut Betriebswirtschaftslehre im Fachbereich Rechts- und Wirtschaftswissenschaften der Technischen Universität Darmstadt, ein funktionsfähiger Forschungs-Prototyp entwickelt. Konzeption und Implementierung wurden von Jörg Kalkmann und dem Autor der vorliegenden Arbeit gemeinsam zur Untersuchung der ***Methode der semantischen Komposition*** und zur Untersuchung eines Repository-Ansatzes für die Verwaltung von Creatorelementen realisiert. Anhand des eingeführten Beispiels wird der Aufbau eines Ordnungssystems für Anwendungselemente in ***TKMS*** erläutert. Anschließend wird eine Variantenkonstruktion und die Komposition einer Beispiel-Anwendung vorgeführt.

4.1.1 Aufbau des Systems

Für ***TKMS*** wurde als Entwicklungsumgebung zu Beginn Powerbuilder Version 5.0 und später Version 6.0 von Sybase Inc. (System der vierten Generation) sowie als Datenbank-Management-System SQL Anywhere 5.5 ebenfalls von Sybase Inc. eingesetzt. Für eine beschleunigte Entwicklung wurden Entwicklungsobjekte der Powerbuilder Foundation Classes-Bibliothek verwendet.

Ein Komponentenverwaltungssystem dient dem geordneten Ablegen und Wiederfinden von Anwendungselementen und Creatorelementen zur Konstruktionszeit, nicht zur Laufzeit. ***TKMS*** zeichnet sich durch Flexibilität und Technologieneutralität aus. Alle Arten von in Datenbanken zu speichernden und für den Konstruktionsprozess relevanten Objekte (z.B. Anwendungselemente, Implementierungsvorschriften Komponenten, Dokumente, Diagramme) können mit ihren Merkmalen in diesem Komponentenverwaltungssystem nicht nur katalogisiert, sondern auch physisch hinterlegt werden. Zwischen diesen Objekten können beliebige semantische oder auch

technologische Abhängigkeiten erfasst werden. Die Anzahl der Objektarten oder der Beziehungsarten kann nach Belieben vom Anwender erweitert werden. Die Grundlagen von ***TKMS*** lassen sich am Besten anhand seines semantischen Datenmodells erläutern.

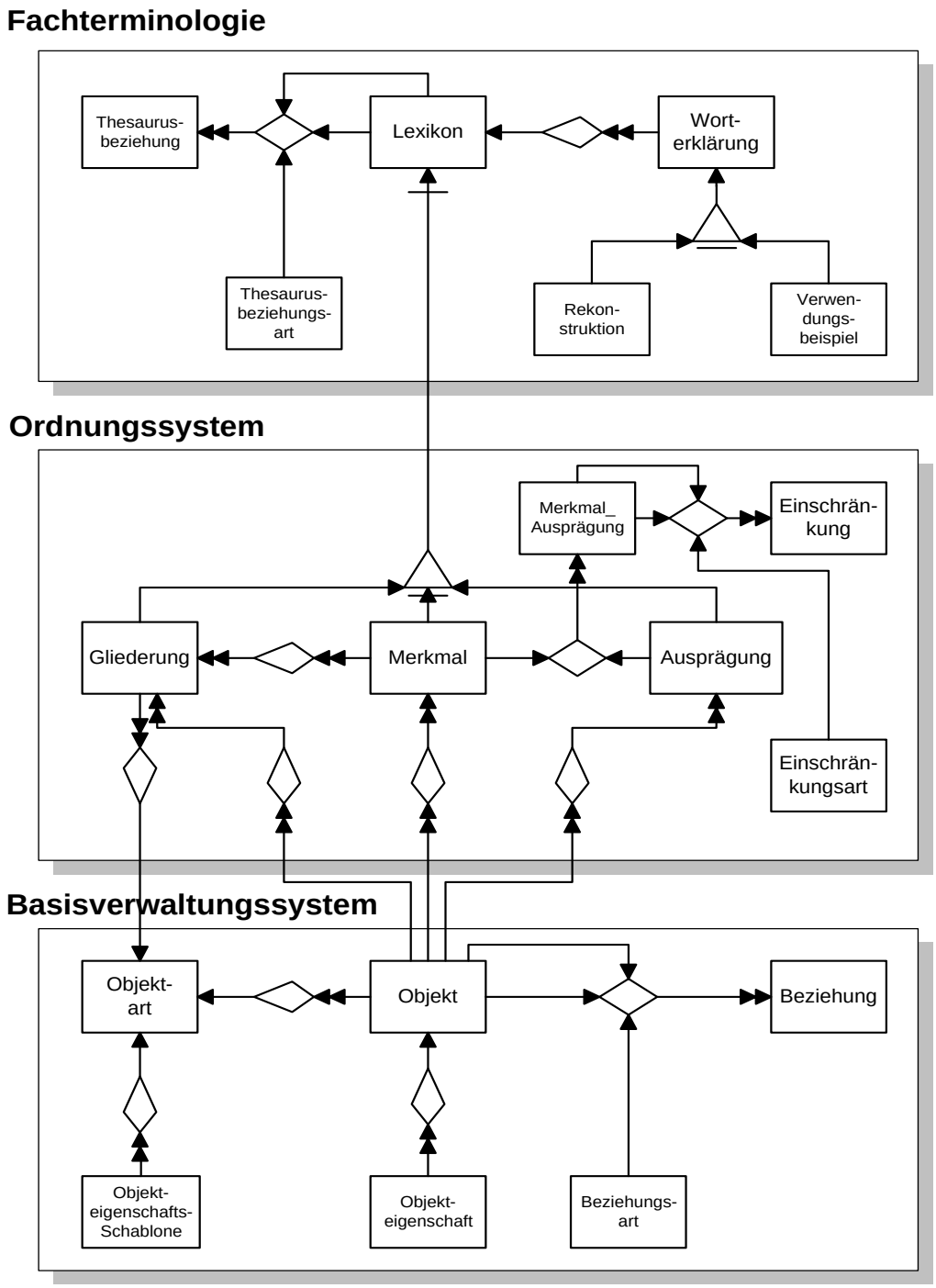


Abbildung 39: Datenmodell des Terminologiebasierten Komponenten-Management-Systems dargestellt in der Objekttypenmethode von Ortner und Söllner

Fachterminologie

Zum Aufbau der Fachterminologie im Komponentenverwaltungssystem werden zwei Teilbereiche identifiziert:

1. Lexikon
2. Thesaurus

Das Lexikon ist eine Sammlung von Fachbegriffen (Termini), die eine Normierung der Fachsprache in den Anwendungsbereichen der Komponenten unterstützen soll. Der interne Aufbau des Lexikons hat Organisations- und Definitionsaspekte [Ortner97, 95ff]. Diesen Aspekten wird durch die Ordnungsfähigkeit des Datenbank-Management-Systems und der Worterklärungen (Rekonstruktion und Verwendungsbeispiel) Rechnung getragen.

Der Thesaurus dient im Komponentenverwaltungssystem zum einen der Definition der Termini über ihre Beziehungen zu anderen Fachbegriffen, zum anderen wird über das Thesaurusbeziehungsnetz mit seinen semantischen Bedeutungen eine koordinierte Suche nach Termini ermöglicht.

Ordnungssystem

Basierend auf der im System hinterlegten Fachterminologie wird ein Ordnungssystem (siehe auch Abschnitt 3.4) für die Erfüllung folgender Aufgaben des Komponenten-verwaltungssystems aufgebaut:

1. Klassifikation bei der Aufnahme der Objekte in das Verwaltungssystem
2. Fachsprachliche Eigenschaftsbeschreibung der Objekte
3. Bereitstellen von Suchinformationen für das Retrieval
4. Repräsentation des Konstruktionskataloges
5. Darstellung der Eignungsprofile im Konstruktionsprozess (Lösungssuche)

Für die Konstruktion mit Katalogen muss ein Komponentenverwaltungssystem eine Abbildung der Katalogbestandteile Gliederung, Hauptteil und Zugriffsteil zur Verfügung stellen.

Der Hauptteil entspricht den Objekten des Verwaltungssystems inklusive ihrer verbalen Beschreibungen (Dokumentation) und graphischen Darstellungen (Bilder, Screenshots, Modelle).

Gliederung und Zugriffsteil werden im ***TKMS*** durch Sachmerkmale und Sachmerkmalleisten realisiert. Gegenstandsgruppen (Gliederungseintrag) werden durch bestimmte Merkmale definiert (Sachmerkmalleiste). Jedem Merkmal wird eine bestimmte Anzahl an Ausprägungen zugeordnet. Jeder Fachbegriff für Gliederung, Merkmal und Ausprägung muss bereits im Lexikon (rekonstruiert) vorhanden sein.

Der Aufbau des Ordnungssystems ist so zu gestalten, dass jedes Objekt in eine Gegenstandsgruppe eingeordnet und durch die Ausprägungen der jeweiligen Merkmale in seinen Eigenschaften beschrieben werden kann. Objekte einer Gegenstandsgruppe, die mit genau denselben Ausprägungen beschrieben werden, stellen eine gleichwertige Lösung für die Teilaufgabe dar und dürfen nicht in ***TKMS*** gespeichert werden, denn aus Sicht des Konstruktionskataloges sind sie redundant.

Ein Suchprofil (Anforderungsprofil) für Objekte, bestehend aus Sachmerkmalen und deren Ausprägungen, für welches kein geeignetes Objekt gefunden wird, stellt schon die Spezifikation für ein neu zu entwickelndes Objekt dar (z.B. Anwendungselement mit neuen Eigenschaften).

Basisverwaltungssystem

In der Tabelle „Objekt“ werden die Objekte physisch aufbewahrt. Die möglichen Typen werden in der Tabelle „Objektart“ hinterlegt. Die Standardeigenschaften eines Objekttyps werden in der Tabelle „Objekteigenschafts-Schablone“ gespeichert und als Vorgaben beim Einfügen neuer Objekte vorgegeben. Die tatsächlichen Eigenschaftswerte eines Objektes werden dann in der Tabelle „Objekteigenschaft“ gespeichert. Beziehungen zwischen beliebigen Objekten sind in der Tabelle „Beziehung“, typisiert nach Beziehungsarten, hinterlegt.

4.1.2 TKMS Beispielanwendung

Die Beispielanwendung für das ***TKMS***-Projekt besteht aus folgenden Anwendungselementen:

1. Kunden-Stammdatenverwaltung mit zwei Varianten: Firmenkunden und Privatkunden
2. Lieferanten-Stammdatenverwaltung
3. Kundenbestellung mit zwei Varianten: Lagerbestellung und Streckenbestellung
4. Lagerbestandsverwaltung
5. Bonitätsprüfung des Kunden
6. Kundenrechnung mit zwei Varianten: Firmenkundenrechnung und Privatkundenrechnung
7. Lieferantenrechnung
8. Offene Posten-Verwaltung für Kunden

Für jedes dieser Anwendungselemente werden mehrere Objekte im Basisverwaltungssystem angelegt. Creatorelemente werden als *.pbd-Dokumente (Powerbuilder Komponenten) mit der Objektart „Creatorelement“ abgelegt. Die Dokumentation für das Creatorelement wird ebenfalls als Objekt mit der Objektart „Dokumentation“ gespeichert und zu dem Creatorelement in Beziehung gesetzt. Jedes Anwendungselement selbst wird auch als Objekt von der Art „Anwendungselement“ angelegt. Anwendungselemente stehen zum einen zu ihren Creatorelementen in einer Beziehung der Art „besteht aus“ und zu anderen Anwendungselementen entsprechend ihrer semantischen Relationen. Beispielsweise wird für die Anwendungselemente *Lagerbestellung* und *Lagerbestandsverwaltung* eine Beziehung der Art „Implikation“ hinterlegt, zwischen den Anwendungselementen *Firmenkundenrechnung* und *Privatkundenrechnung* wird eine Beziehung der Art „Äquivalenz“ definiert und von der *Streckenbestellung* hin zur *Lagerbestandsverwaltung* wird eine Beziehung der Art „Negation“ festgelegt. Für die Baugruppenbildung zwischen der *Firmenkunden-Stammdatenverwaltung* und der *Firmenkundenrechnung* wird eine Beziehung der Art „besteht aus“ gespeichert.

Mit den Objektarten und Beziehungsarten ist das TKMS in der Lage im Variantenkonstruktionsprozess bei der Lösungssuche die Anwendungselemente und ihre Baugruppen herauszufinden. Zudem können die Beziehungen zwischen den Anwendungselementen für die Zusammenstellung der Gesamtlösung entsprechend ihrer Beziehungsarten eingesetzt werden (Vorschlag oder Konfliktwarnung).

Eine fertige Anwendung sieht folgendermaßen aus[48]:

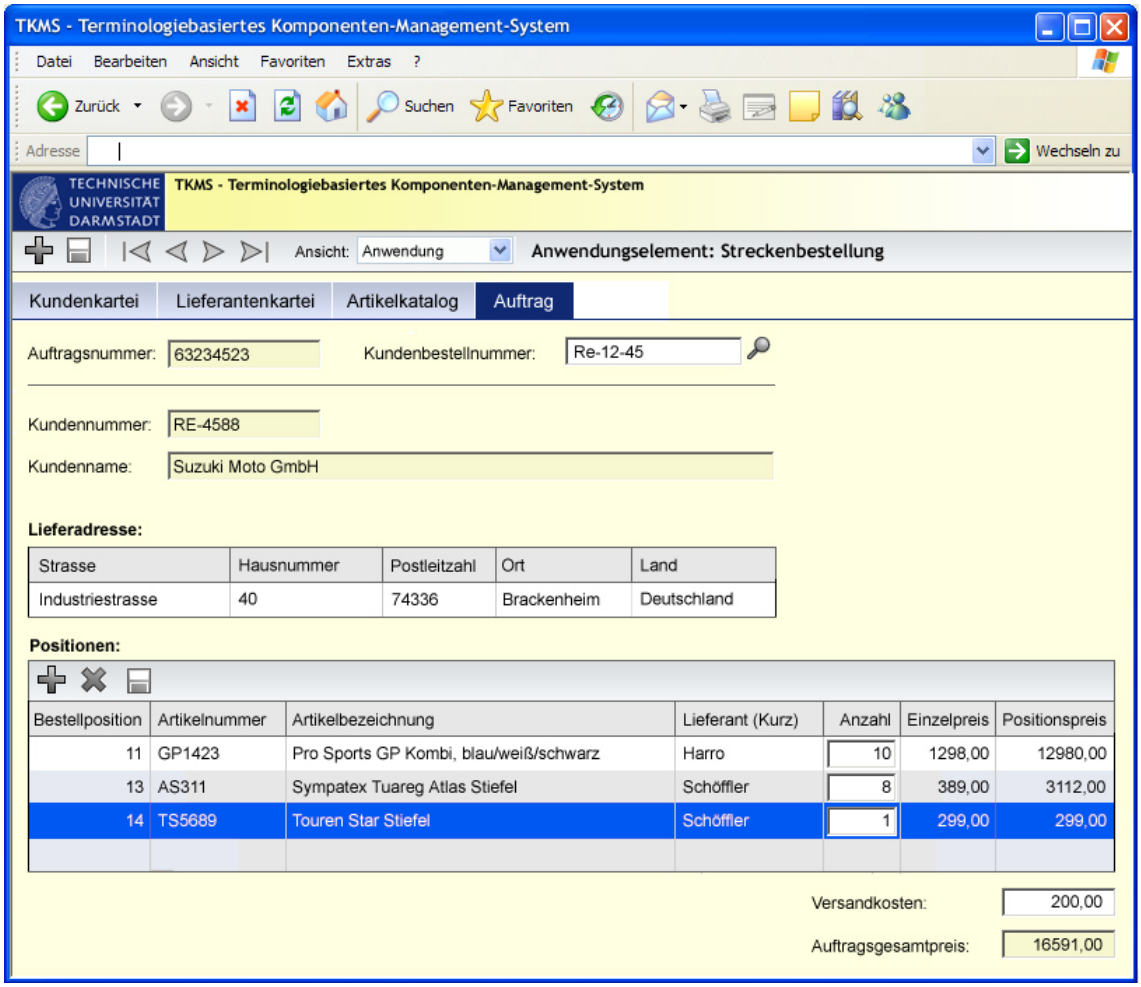


Abbildung 40: Benutzeroberfläche des Anwendungselementes *Streckenbestellung*

Nach der Komposition der Gesamtlösung wird die Implementierungsvorschrift ausgeführt. Mit ***TKMS*** werden Browser Plug-Ins erzeugt, damit die zusammengestellten Anwendungselemente als Gesamtanwendung im Browser bedient werden

[48] Die aufgeführten Bildschirmdarstellungen wurden optisch an aktuelle User Interaction Styleguides angepasst, sie entsprechen funktional jedoch genau den ursprünglichen Benutzeroberflächen von ***TKMS***.

können. Das User Interface für die Benutzerinteraktion ist Teil der Creatorelemente. Die Menüführung wird bei der Komposition in ***TKMS*** als einfache „Tab-Reiter"-Reihenfolge mit Link auf die Anwendungselemente definiert und als HTML-Datei in ein Verzeichnis installiert.

TKMS kann in zwei verschiedenen Ansichten arbeiten: Anwendung oder Repository. Für die Anwendung erzeugt TKMS eine Tab-Reiter-Struktur, wie in diesem Beispiel: Kundenkartei, Lieferantenkartei, Artikelkatalog und Auftrag. Die Namen der Menüeinträge sind in jedem Projekt frei wählbar. Für die Auftragsbearbeitung werden hier drei Anwendungselemente mir ihren Benutzeroberflächen dargestellt. Die *Streckenbestellung* (Abbildung 40) ist eine Variante der *Lagerbestellung* (Abbildung 41), d.h. die beiden Anwendungselemente sind austauschbar. Die *Lagerbestellung* muss mit der *Lagerverwaltung* (Abbildung 42) zusammen installiert werden. Zwischen *Lagerbestellung* und *Lagerverwaltung* gibt es eine Implikationsbeziehung. Für den Fall, dass die *Lagerbestellung* mit der *Lagerverwaltung* als Baugruppe fest zusammengebaut werden soll ist auch noch die Baugruppe „Lagerverwaltung mit Bestellung" als Objekt im Repository anzulegen und mit der „besteht aus"-Beziehung zu den beiden elementaren Anwendungselementen zu versehen.

Kundenbestellung mit Streckenauftrag bedeutet, dass die Lieferung nicht aus dem Lager des Verkäufers erfolgt, sondern ein Lieferauftrag an den Warenlieferanten des Verkäufers geht, damit dieser die Lieferung direkt zum Kunden (Besteller) versendet.

Bei der Auftragserfassung werden die Kunden aus den Kundenstammdaten selektiert (ausgewählt). Nur Kunden, die zuvor in der Kundenkartei erfasst wurden, sind für die Auswahl zugelassen. In der Bestellposition werden Artikelnummern eingegeben. Ist die Nummer nicht korrekt oder nicht bekannt, wird über einen Suchdialog im Artikelkatalog nach dem Artikel gesucht. Für die Lieferung muss noch ein geeigneter Lieferant ausgewählt werden. Die Auswahl wird durch einen Suchdialog unterstützt. Es sind nur Lieferanten aus der Lieferantenkartei zugelassen.

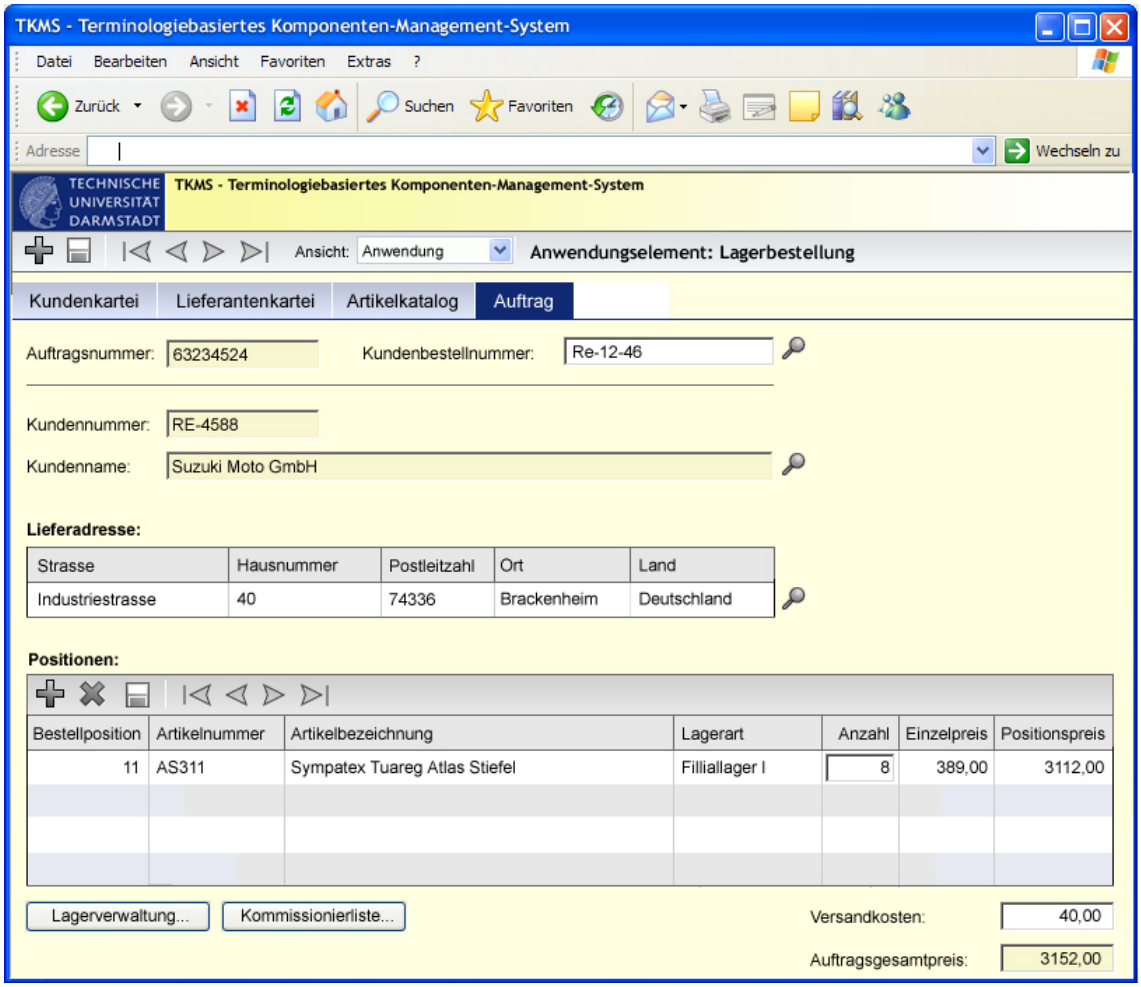


Abbildung 41: Benutzeroberfläche des Anwendungselementes *Lagerbestellung*

Hat der Verkäufer seine Artikel selbst auf Lager so kann das Anwendungselement Lagerbestellung eingesetzt werden. Die Kundendaten werden ebenfalls aus der Kundenkartei selektiert. Allerdings werden hier keine Lieferantendaten der Artikelposition zugeordnet, sondern es wird nach einem geeigneten Lagerort für die Entnahmen gesucht. Mit der Schaltfläche „Lagerverwaltung“ kann bei diesem Anwendungselement die Lagerverwaltung geöffnet werden.

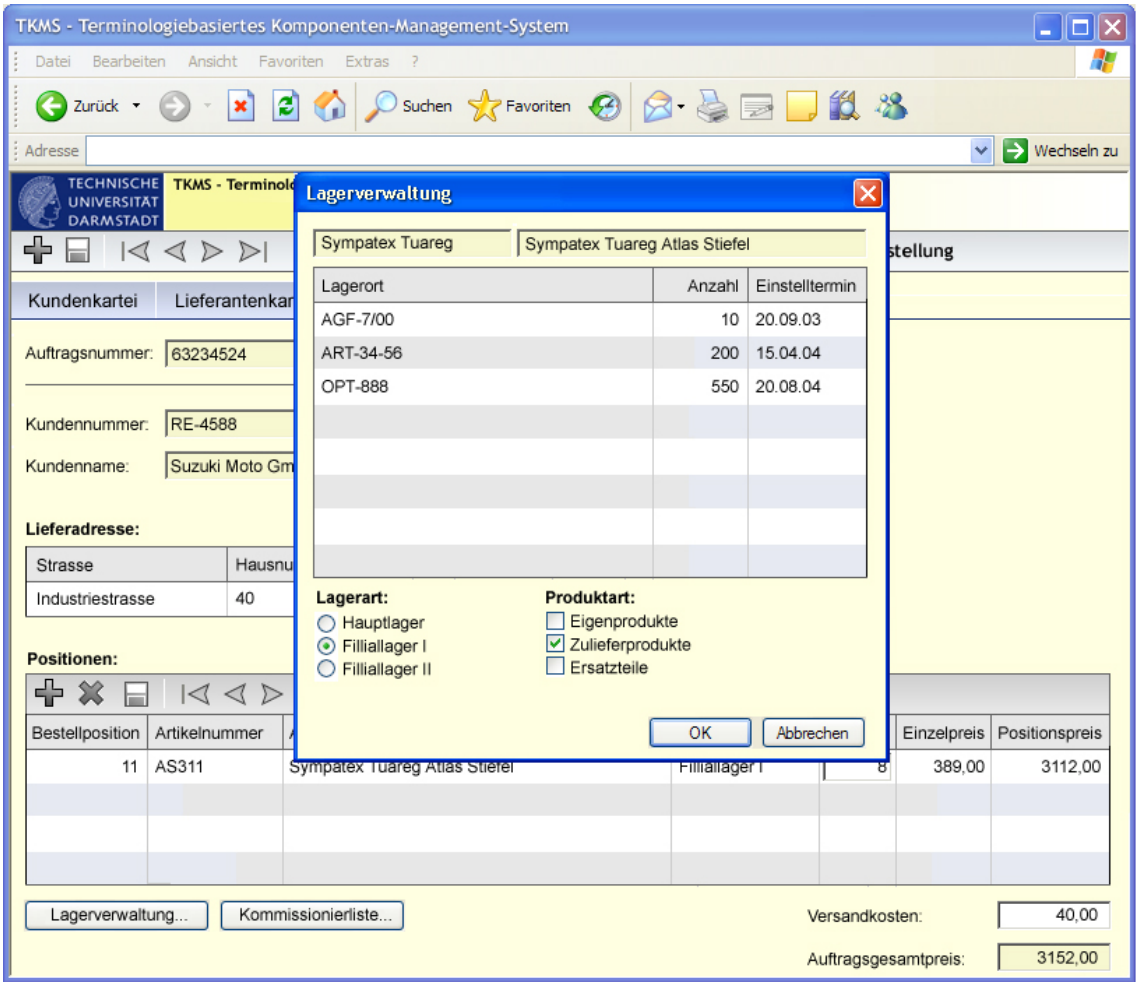


Abbildung 42: Benutzeroberfläche des Anwendungselementes *Lagerbestellung* mit geöffnetem Dialogfenster für Lagerinformation

In der Lagerverwaltung kann in verschiedenen Lägern die Verfügbarkeit des Artikels nachgesehen werden. Der passende Lagerort wird ausgewählt und steht dann in der Kommissionierliste für die Entnahme als Information.

4.1.3 Übergang zwischen Anwendungselement und der Implementierung seines Creatorelementes

Für das Anwendungselement *Lagerbestellung* werden im Folgenden die Sachmerkmale und die Implementierungsvorschrift erläutert sowie die Implementierung des referenzierten Creatorelementes dargestellt. Im Projekt ***TKMS*** wurden DCOM-Komponenten in Powerbuilder generiert. Für eine bessere Darstellung ist hier statt der

DCOM-Komponente *Lagerbestellung* eine Java-Implementierung mit gleicher Funktionalität skizziert[49].

Der fachliche Anteil des Anwendungselementes „Lagerbestellung“ ist in ***TKMS*** feiner detailliert als in Abbildung 18 gezeigt. Die Eigenschaft *Produktlieferung* ist eigentlich eine komplexe Eigenschaft[50] und wird in ***TKMS*** durch mehrere Merkmale und Ausprägungen abgebildet. Folgendes Gegenstandsmuster für explizite Eigenschaften (siehe Abschnitt 3.4.2) ist für die *Lagerbestellung* im Repository hinterlegt:

Merkmale		**Ausprägungen**
Bestandsführung	–	Bestandsgeführt
Produktbereitstellung	–	Kommissionierung
Rahmenvereinbarung	–	Auftragsrabatt
Terminierung	–	Spezieller Liefertermin
Arbeitsteilung	–	Zentralisierte Bearbeitung
Kundenart	–	Firmenkunden

Tabelle 15: Gegenstandsmuster für die *Lagerbestellung* in ***TKMS***

Die folgende Grafik zeigt ein UMLKlassendiagramm für die Dokumentation des Creatorelementes *Kundenbestellung.Lager*.

[49] Mit dem Modellierungswerkzeug Together wurde das UML-Diagramm und mit der Entwicklungsumgebung Eclipse das Coding auf Basis der DCOM Vorlage von Georg Wilhelm erzeugt.

[50] Die Problematik und Handhabung von komplexen Eigenschaften wurde in Abschnitt 3.4.2 behandelt.

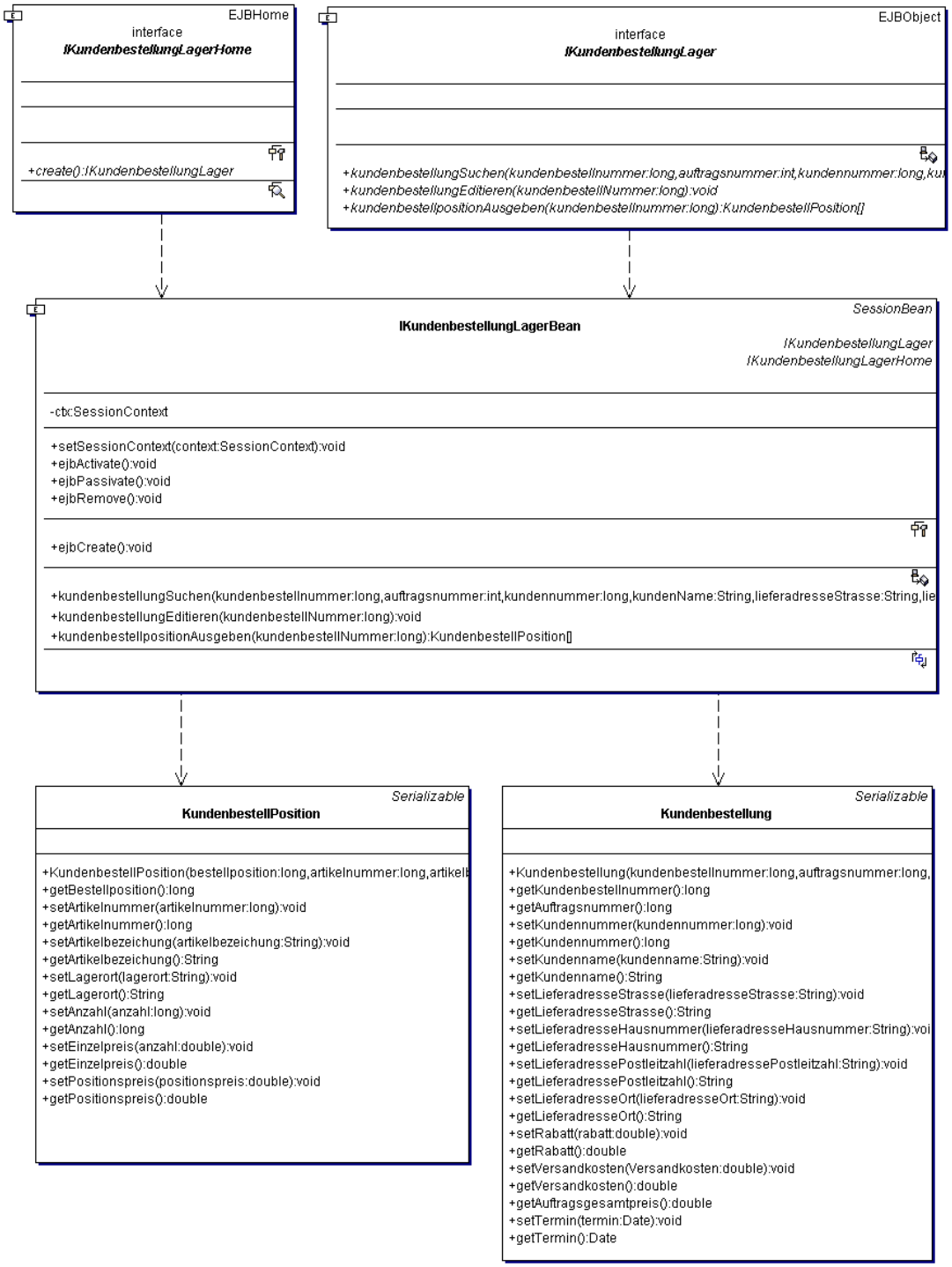


Abbildung 43: UML-Klassendiagramm für Creatorelement Kundenbestellung.Lager

Das Creatorelement wird als EJB implementiert. Zwei einfache Java-Objekte „KundenbestellPosition“ und „Kundenbestellung“ übernehmen das Lesen und Schreiben der Daten in die Datenbank. Typischerweise haben EJB’s zwei Interfaces, ein Home Interface und ein Object Interface. Ein Java Client ruft über den Java Naming Directory Interface Service (JNDI Service) mit dem logischen Namen die Komponente

und JNDI referenziert dann auf das Home Interface. Anschließend kann der Client das Object Interface ansprechen. Das Object Interface repräsentiert die physikalische Implementierung der Anwendungselemente Lieferanten-Schnittstelle.

Das nachfolgend dargestellte Coding zeigt die Implementierung des Datenbankzugriffs in JDBC dargestellt für das Java-Objekt *KundenbestellPosition*:

```
public KundenbestellPosition[] kundenbestellpositionAusgeben
    (long  kundenbestellNummer)
{
   ...
   ArrayList kundenbestellPositionenList = new ArrayList();

   String query = "SELECT BESTELLPOSITION ARTIKELNUMMER " +
       "ARTIKELBEZEICHUNG LAGERORT ANZAHL EINZELPREIS POSITIONSPREIS " +
       "FROM KUNDENBESTELLPOSITION WHERE KUNDENBESTELLNUMMER = " +
                kundenbestellNummer +
       " ORDER BY BESTELLPOSITION";
   try
   {
     Statement statement = con.createStatement();
     ResultSet result =  statement.executeQuery(query);
     KundenbestellPosition bestellPosition = null;
     while(result.next())
     {
       bestellPosition = new  KundenbestellPosition(result.getLong("BESTELLPOSITION"),
          result.getLong("ARTIKELNUMMER"),
          result.getString("ARTIKELBEZEICHNUNG"),
          result.getString("LAGERORT"),
          result.getLong("ANZAHL"),
          result.getDouble("EINZELPREIS"),
          result.getDouble("POSITIONSPREIS"));
       kundenbestellPositionenList.add(bestellPosition);
     }
   }
   catch(SQLException exception)
   {}

   KundenbestellPosition[] kundenbestellPositionen =
          new KundenbestellPosition[kundenbestellPositionenList.size()];
   Iterator iter = kundenbestellPositionenList.iterator();
   int i = 0;
   while(iter.hasNext())
   {
    kundenbestellPositionen[i] =
          (KundenbestellPosition)iter.next();
    i++;
   }
   return kundenbestellPositionen;
}
```

Abbildung 44: Datenbankzugrif für das Java Objekt *KundenbestellPosition*

Für die physische Installation werden im Basisverwaltungssystem von ***TKMS*** zwei Dateien gespeichert (oder referenziert). Das ist einerseits KundenbestellungLagerBean.jar, eine komprimierte Datei und andererseits die zugehörige descriptor.xml-Datei, für die zum Installationszeitpunkt noch der logische Datenbankname durch den physischen Datenbanknamen des installierten Systems ersetzt werden muss.

Die Implementierungsvorschrift für das Anwendungselement *Lagerbestellung* wird nun folgendermaßen abgearbeitet. *Lagerbestellung* hat eine „besteht aus“-Beziehung zum Creatorelement *Kundenbestellung.Lager*. Diesem Creatorelement ist als Eigenschaft hinterlegt, dass es ein Enterprise Java Bean ist. Weiterhin hat „Kundenbestellung.Lager“ eine „besteht aus“-Beziehung zu der Installationsdatei KundenbestellungLagerBean.jar und diese wiederum eine Beziehung zu ihrer descriptor.xml-Datei. Für Creatorelemente mit der Eigenschaft EJB installiert das Komponentenverwaltungssystem die *.jar und *.xml Dateien einer gesamten Anwendung in eine weitere Zip-Datei *.ear und installiert die finale Datei dann auf einem J2EE Application Server.

Zusammengefasst bedeutet dies, dass entsprechend der Komposition von Anwendungselementen zu einer Gesamtlösung die Implementierungsvorschriften ausgeführt werden und für Creatorelemente mit der Eigenschaft EJB werden die entsprechenden Installationsdateien erzeugt und an die geeignete Stelle verteilt.

Beispiel für Kunden- und Lieferanten-Schnittstellen

In Ergänzung zu Abschnitt 3.3.8.1 lässt sich die Kommunikationsproblematik für Schnittstellen bei einem Austausch von Anwendungselemente-Varianten an dem hier angeführten Beispiel gut erklären. Angenommen, das Anwendungselement *Lagerbestellung* soll gegen ein Anwendungselement *Streckenbestellung* ausgetauscht werden, so gilt:

AE *Lagerbestellung* hat die Lieferanten-Schnittstelle [L(kundenbestellpositionAusgeben)] anzubieten. Diese wird zum einen von einem Anwendungselement für die Lagerverwaltung genutzt (AE *Lagerverwaltung*) und zum anderen von einem Anwendungselement für die Erstellung von Rechnungen (AE *Privatkundenrechnung*). Somit haben die beiden letztgenannten Anwendungselemente also je eine Kunden-

Schnittstelle [K(kundenbestellpositionErmitteln)], die von [L(kundenbestellpositionAusgeben)] bedient wird.

Soll nun AE *Lagerbestellung* gegen seine Variante AE *Streckenbestellung* ausgetauscht werden, so muss die *Streckenbestellung* als Anwendungselemente-Variante auf jeden Fall die Lieferanten-Schnittstelle [L(kundenbestellpositionAusgeben)] besitzen, damit AE *Lagerverwaltung* und AE *Privatkundenrechnung* unter Verwendung von [K(kundenbestellpositionErmitteln)] weiterhin erfolgreich mit der Anwendungselemente-Variante über diese Lieferantenschnittstelle kommunizieren können. Genau dann sind *Lagerbestellung* und *Streckenbestellung* bezüglich ihrer Lieferanten-Schnittstellen äquivalent.

Übersicht weiterer Möglichkeiten für die Ablage von Implementierungsvorschriften in *TKMS*

Wie in Abschnitt 3.2.1 bereits beschrieben, können BSP-basierte Standard-Anwendungssysteme aus unterschiedlichen Komponentenausprägungen aufgebaut werden. Softwarekomponenten und Composite Applications werden je nach Technologie, ähnlich wie das oben angeführte EJB-Beispiel in einem Komponentenverwaltungssystem abgelegt und über die Ausführung der jeweiligen Implementierungsvorschrift installiert. Bei den anderen Komponentenausprägungen kommt es sehr auf die technische Umsetzung der jeweiligen Komponenten an (z.B. Formulareditor, Portaladministration, EAI-Werkzeug für B2B-Schnittstellen).

Ein Ergebnis der vorliegenden Forschungsarbeit ist, dass die Aktivierung der Komponenten durch die Übergabe einer XML-Datei (Konfigurationsdatei) mit Inhalten und Formaten, passend zu den Realisierungstechniken erfolgen sollte. Das bedeutet beispielsweise, dass die Implementierungsvorschrift für ein Anwendungselement *Kommissionierliste* aus einer Konfigurationsdatei für die Konfiguration der *Lagerbestellung* und einer Konfigurationsdatei für das Formular *Kommissionierliste* besteht. Alternativ würde das Programmcoding für eine Creatorelement-Komponente für die Kommissionierung aktiviert werden und die Formulardatei für die Kommissionierliste müsste installiert werden. Voraussetzung für eine Aktivierung durch Konfigurationsdateien ist, dass die Realisierungstechnik generisch (multifunktional) entwickelt wurde und auf die Konfigurationsdaten entsprechend reagieren kann. Diese Vorgehensweise wird dann auf Konfigurations-Parameter-Pakete, Portal-Pakete und

Benutzerschnittstellen sowie für Formulare und Berichte und auch für B2B-Schnittstellen angewandt.

Mit dieser Vorgehensweise wird das gewünschte Implementierungsergebnis von der Programmverwaltung entkoppelt. Halten die Realisierungstechniken ihre Schnittstellen für den Empfang der Konfigurationsdateien stabil, so hat die XML-Datei, welche die Implementierungsvorschrift von Anwendungselementen enthält, ebenfalls eine lange Lebensdauer. Dadurch, dass die Konfigurationsdateien ausschließlich durch Anwendungselemente in einem Konstruktionsprozess repräsentiert werden, ist das Prinzip der Wiederverwendung durch Auswahl (siehe Abschnitt 3.3.6) vollständig eingehalten. Mit diesem Konzept ist es möglich den Release-Wechsel (Upgrade) von Standard-Anwendungssystemen zu unterstützen. Zum einen ist dem Komponentenverwaltungssystem auf der Seite des Kunden-Unternehmens bekannt, welche Anwendungselemente bei diesem Kunden aktiv sind. Damit kann eine notwendige Änderung der Implementierungsvorschrift genau identifiziert und an den Kunden kommuniziert werden. Zum anderen können die Werkzeuge, welche die XML-Dateien empfangen, ihre inneren Strukturen weiterentwickeln und dennoch zu den bereits an Kunden ausgelieferten Konfigurationsdaten kompatibel bleiben (Abwärtskompatibilität).

4.1.4 Datenablage für das Anwendungselemente-Beispiel in Ordnungssystem und Fachterminologie von TKMS

In ***TKMS*** wird ein Anwendungselement zuerst im Basisverwaltungssystem aufgenommen und in sein Beziehungsnetz eingebettet, d.h. alle seine bekannten Beziehungen zu andern Objekten im Basisverwaltungssystem werden angelegt. Anschließend muss das Anwendungselement auch in das Ordnungssystem eingeführt werden, andernfalls steht es für Konstruktionshandlungen nicht zur Verfügung. Das bedeutet, dass Anwendungselemente, die nicht mit Merkmalausprägungen versehen sind und somit nicht über ihre Sachmerkmalleiste in die Gliederung des Konstruktionskataloges eingeordnet wurden, aus der Sicht einer Variantenkonstruktion nicht existieren.

Demzufolge muss jeder, der das Baukastensystem erweitern will, nicht nur die Implementierungsvorschrift für ein Anwendungselement erstellen, sondern auch die Ergebnisse des sprachbasierten Fachentwurfs in dem Komponentenverwaltungssystem ablegen. Erst damit wird sichergestellt, dass für die Auswahl und Komposition

von Anwendungselementen in einem Variantenkonstruktionsprozess die Sprache und Gliederungsstruktur des Anwendungsbereiches zur Verfügung stehen.

Ordnungssystemaufbau

Für das Beispiel der Lagerbestellung müssen auf Schema-Ebene die Merkmale *Bestandsführung, Produktbereitstellung, Rahmenvereinbarung, Terminierung, Arbeitsteilung* und *Kundenart* angelegt werden. In der Gliederung wird für diese Merkmale eine Sachmerkmalleiste *Kundenbestellung* als unterste Ebene (Blattknoten) angelegt. Über diese Sachmerkmalleiste werden die Merkmale auf der Schema-Ebene gruppiert. Innerhalb der Gliederung stellt eine konkrete Sachmerkmalleiste eine Instanz für die Repräsentation einer Gruppe von Lösungsvarianten für die Teilaufgabe „Kundenbestellung“ dar.

Anschließend werden die Ausprägungen der jeweiligen Merkmale als Schema vollständig angelegt (z.B. Merkmal: *Rahmenvereinbarung* – Ausprägungen: *Auftragsrabatt, Mengenkontrakt, Wertkontrakt*).

Ist die Gliederungshierarchie oberhalb der Sachmerkmalleiste noch nicht ausgeprägt, d.h. die Sachmerkmalleiste kann nicht in eine bestehende Strukturierung des Anwendungsbereiches eingeordnet werden, so muss die Hierarchie entsprechend den Ergebnissen des komponentenorientierten Systementwurfs aufgebaut werden.

Diese Vorgehensweise unterscheidet sich nicht für elementare Sachmerkmalleisten und Sachmerkmalleisten für Baugruppen.

Erst nach Abschluss des Schemaaufbaus für das Ordnungssystem kann die Instanz eines Anwendungselementes in dieses Ordnungssystem eingeführt werden. Dafür werden die jeweiligen Merkmalausprägungen in Form eines Gegenstandsmusters dem Anwendungselement zugeordnet. Das bedeutet, dass jedes Anwendungselement für [alle bzw. einige] Merkmale genau eine Sachmerkmalleiste [eine bzw. mehrere] Ausprägungen besitzt. Freiheitsgrade für die Zuordnung entstehen, wenn Multigegenstände und Nullzustände zugelassen werden. In ***TKMS*** werden Einschränkungen durch Einschränkungsbeziehungen zwischen Merkmalausprägungen abgebildet und im Programm ausgewertet.

Jeder Fachbegriff für Gliederungselemente, Merkmale und Ausprägungen muss vor der Verwendung im Ordnungssystem bereits im Lexikon definiert sein.

Aufbau der Fachterminologie

Bevor ein Fachbegriff in das ***TKMS***-Lexikon aufgenommen werden kann, muss erst eine Rekonstruktion der Fachbegriffe erfolgen. Für die Rekonstruktion und Klärung der Anwendungsbereichsfachsprache werden die Methoden und Vorgehensweisen der sprachbasierten Anwendungsentwicklung (siehe Abschnitt 2.3) eingesetzt. Das Lexikon in einem Komponentenverwaltungssystem für betriebswirtschaftliche Standard-Anwendungssysteme sollte sich, wenn vorhanden, auf international anerkannte Begriffsdefinitionen abstützen. Der weitergehende Aufbau des Wortschatzes wird durch folgenden Prozess begleitet[51]:

1. Sammlung der Aussagen von Fachexperten, die einen Anwendungsbereich mittels Fachbegriffen beschreiben.
2. Aussagen werden normiert, indem sie in normierte Satzbaupläne überführt werden und ihre Fachbegriffe möglichst durch gleichwertige bereits normierte Fachbegriffe aus dem Lexikon ersetzt werden (mit Referenz auf den ersetzten Bezeichner).
3. Sind Fachbegriffe nicht ersetzbar, so werden auf der Basis der normierten Aussagen die Fachbegriffe rekonstruiert, indem sie explizit definiert, mit Beispielen und Gegenbeispielen präzisiert und in ein Beziehungsnetz zu anderen Fachbegriffen eingeordnet werden (Thesaurusbeziehungen).

Die Ausführungen in Abschnitt 3.4.6 erläutern den grundsätzlichen Aufbau eines Lexikons und seiner Thesaurusbeziehungen. In TKMS werden Begriffe diesem Prinzip folgend in das Lexikon eingetragen. Das Beispiel aus Tabelle 13 stellt einen Eintrag mit Worterklärung und Verwendungsbeispiel dar.

[51] Hier wird nur die Rekonstruktion der Fachbegriffe betrachtet. Für einen Fachentwurf für Anwendungselemente muss die Aussagennormierung, d.h. die Erzeugung einer Normsprache (Lexikon und Grammatik) vollständig durchgeführt werden.

Die Ergebnisse der Fachbegriffsrekonstruktion und die Ergebnisse der Anwendungsbereichsgliederung sowie die Ergebnisse des methodenspezifischen Systementwurfs von Anwendungselementen und ihren semantischen Beziehungen sowie ihren Implementierungsvorschriften stehen somit in ***TKMS*** für die Variantenkonstruktion zur Verfügung.

4.2 Durchführung der Variantenkonstruktion

In einem Variantenkonstruktionsprozess ist die Auswahl der Lösungsvarianten (aus dem Konstruktionskatalog) nur ein Schritt von vielen. Zuvor müssen die Anforderungen erfasst und daraus die Gesamtaufgabe abstrahiert werden. Anschließend erfolgt die Festlegung der Teilaufgaben. Bei der Suche nach den geeigneten Teilaufgaben sind neben den Referenzplänen einer Branche auch die Erfahrungen früherer Projekte zu verwenden, die in unterschiedlichster Weise im System hinterlegt bzw. durch geeignete Werkzeuge bereitgestellt werden können. Weiterhin sollten verschiedene aspektemodellierende Vorgehensweisen und deren Ergebnisse in den Konstruktionsprozess mit einbezogen werden (siehe Abschnitt 3.3.2). Unter Aspekten sind Geschäftsprozesse, Arbeitsabläufe, Berechtigungskonzepte, Stammdaten-Analysen und weitere zu verstehen (Filtersysteme). Die Werkzeuge zur Aspektemodellierung sollten mit einem Composer-Tool integrativ zusammenwirken, denn bei der Auswahl der Komponenten für ein Konstruktionsprojekt gibt es viele sich ergänzende und gleichwertige Sichtweisen auf dieselben Probleme [Lang98, 24].

Bei der Erstellung eines Aufgabenplans fängt man beispielsweise mit der Geschäftsprozessmodellierung an, aus der die Arbeitsschritte – dies können Handgriffe (physische Arbeit) und/oder Sprachhandlungen (geistige Arbeit) sein – durch weitere abstraktive und kompositive Zerlegung der Prozesse (Vorgänge) ermittelt werden. Aus den einzelnen Arbeitsschritten kann man dann Aufgaben aggregieren, die Arbeitspersonen bzw. ihren Stellen zugeordnet werden können. Aufgabe eines „Efficiency Engineering“ ist es hierbei, die unter verschiedenen Randbedingungen (Fähigkeiten der Mitarbeiter, Funktionalität der Arbeitsmittel, Nebenläufigkeit der Teilarbeitsschritte etc.) optimale Verteilung der Arbeit zu ermitteln. Auf dieser Stufe lassen sich dann eine Stellenstruktur, die Zusammenfassung von Stellen zu Organisationseinheiten und daraus - bezogen auf die Elemente einer Aufbau-

organisation (z.B. Verteilungsaspekt) - ein Aufgabenplan entwickeln, der die Grundlage für eine Komposition von Anwendungselementen zu Anwendungssystemen (Arbeitsmitteln) bildet [Ortner et.al. 99b].

Die Modellierung von Ablauf- und Aufbauorganisation muss in Kooperation mit den Konstruktionshandlungen der Variantenkonstruktion erfolgen. Variantenkonstruktion bedeutet auch, dass Anforderungen in einen Aufgabenplan überführt werden, indem der Konstrukteur die Ordnungsstrukturen der Abstraktions- und Kompositionshierarchie als Gliederung des Konstruktionskataloges für das Denken in Baugruppen und das Finden von funktionalen und prozessualen Teilaufgaben einsetzt.

Eine Teilaufgabe korreliert im Idealfall mit einer Sachmerkmalleiste. Oder umgekehrt gilt, Sachmerkmalleisten müssen so definiert werden, dass sie die Lösungssortimente für Teilaufgaben repräsentieren. Durch die Kongruenz von Sachmerkmalleisten und Teilaufgaben kann die Strukturierung der Gliederung und der Sachmerkmalleisten für das Finden von geeigneten Teilaufgaben genutzt werden. Zudem bietet der kompositive Teil der Gliederung quasi als Referenz-Komposition Vorschläge für ergänzende Teilaufgaben. Weiterhin kann ein Expertensystem zur Verarbeitung der semantischen Relationen in Form von Forderungen an „informative Vollständigkeit“ geeignete Teilaufgaben vorschlagen bzw. bei „Inkompatibilität“ die Bearbeitung anderer Teilaufgaben ablehnen.

Ist die Variantenkonstruktion soweit fortgeschritten, dass die Teilaufgaben feststehen, sind nur noch die beiden Konstruktionshandlungen Auswahl und Komposition zugelassen, um aus bestehenden Bausteinen eines Baukastensystems Anwendungssoftware herzustellen. Die Auswahl basiert rein auf Fachbegriffen und Inhalten des Anwendungsbereichs der Software. Die Komposition verwendet ausschließlich Konstruktionselemente (Anwendungselemente), die für den Konstrukteur (Composer) inhaltlich definiert sind.

Eine schematische Darstellung eines Variantenkonstruktionsvorganges mit Anwendungselementen wird in der folgenden Abbildung gezeigt:

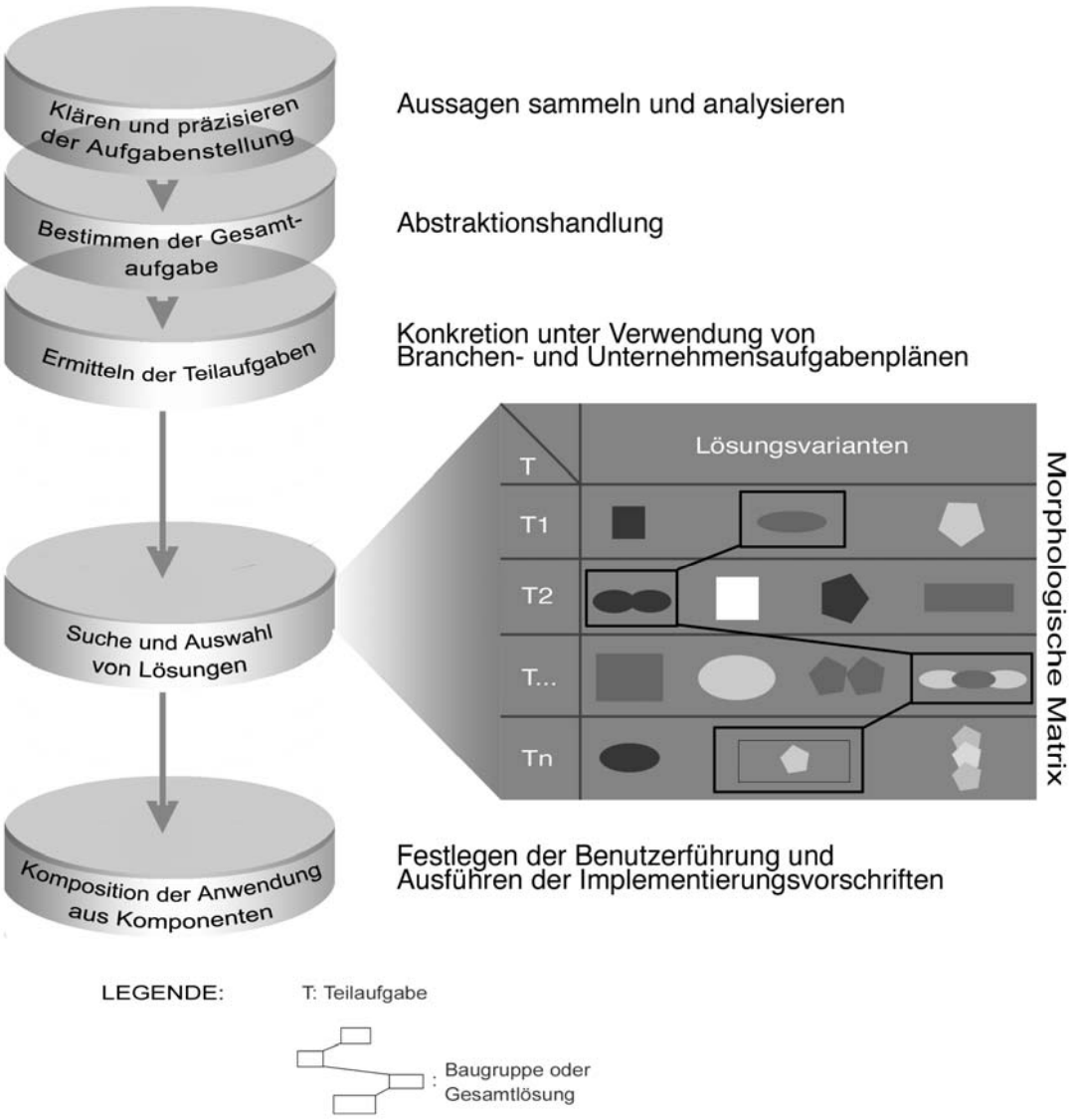


Abbildung 45: Vereinfachter Konstruktionsprozess einer Anwendung aus Komponenten [Ortner et.al. 99b, Lang98, 25]

Eine morphologische Matrix (Synonym: morphologischer Kasten, siehe Abschnitt 2.1.3) ist ein Ordnungsschema, welches die systematische Kombination bekannter Lösungselemente zu einer (neuen) Gesamtlösungen vorantreibt [VDI 2212-81, Pahl/Beitz93]. Ziel dabei ist es, das vorhandene Lösungsspektrum für einzelne Teilaufgaben vollständig und übersichtlich zu präsentieren. In der ersten Spalte werden zeilenweise die zu lösenden Aufgaben aufgeführt und innerhalb dieser Zeilen stehen die dafür möglichen Lösungen. Zur Unterstützung der Lösungsauswahl werden zusätzliche Informationen (z.B. Eigenschaftsbeschreibungen) bereitgestellt.

In dem hier vorgestellten Ansatz gibt es eine universale Morphologische Matrix, die das gesamte Baukastensystem, geordnet nach Teilaufgaben (Sachmerkmalleisten), darstellt. Als Zusatzinformationen werden primär die Merkmalausprägungen der Anwendungselemente verwendet. Für ein konkretes Konstruktionsprojekt werden die relevanten Teilaufgaben ermittelt und die universale Morphologische Matrix entsprechend segmentiert. Demzufolge steht dann eine projektspezifische Matrix zur Verfügung, bei der schrittweise (zeilenweise) zu jeder Teilaufgabe aus der Menge der möglichen Anwendungselemente das geeignete ausgewählt und mit den Anwendungselementen der anderen Teilaufgaben zu einer Gesamtlösung komponiert wird. Die Auswahl wird durch die Definition eines Anforderungsprofils für Anwendungselemente (Kombination aus Merkmalausprägungen) unterstützt.

Ist eine Teilaufgabe abstrakter als vorhandene Anwendungselemente, so kann der Composer selbst eine Anwendungselemente-Baugruppe zur Lösung der Aufgabe erstellen. Stehen einem Anforderungsprofil mehrere gleichwertige Lösungsalternativen gegenüber, so ist das Anforderungsprofil zu verfeinern.

Für einige Teilaufgaben finden sich im Lösungskatalog keine Varianten mehr, weil eine Filterung des Bausteinbestandes stattgefunden hat. Filtersystem-Konfigurationen und die Lösungsauswahl im Konstruktionskatalog erfolgen nicht vollständig sequentiell, es finden immer wieder Interdependenzen statt.

Nach Abschluss einer erfolgreichen Komposition repräsentiert der Anwendungsbauplan die Gesamtheit der ausgewählten Lösungen und ist noch um die Benutzerführung (Menüsteuerung) zu ergänzen. Der Anwendungsbauplan ist die Grundlageninformation für die Ausführung der Implementierungsvorschriften aller in ihm verwendeter Anwendungselemente. Nach erfolgter Implementierung existiert ein lauffähiges den Kundenanforderungen entsprechendes Anwendungssystem.

Die Einbindung eines Anwenders in die Arbeitsablaufkonzepte und Aufbauorganisationsstrukturen seines Unternehmens ist auf der Grundlage der Anwendungselemente neu zu betrachten. Da der Aufgabenplan die Stellenstruktur (Organisation) und das Prozessablaufmodell eines Unternehmens vereinigt, können entlang der Teilaufgaben die notwendigen Arbeitsmittel der einzelnen Mitarbeiter identifiziert und zu einem benutzerspezifischen Arbeitsplatz zusammengesetzt werden [Lang97].

Eine schematische Einordnung benutzerspezifischer Arbeitsplätze mit ihren Komponenten in die Unternehmensmodellierung wird in der folgenden Abbildung skizziert.

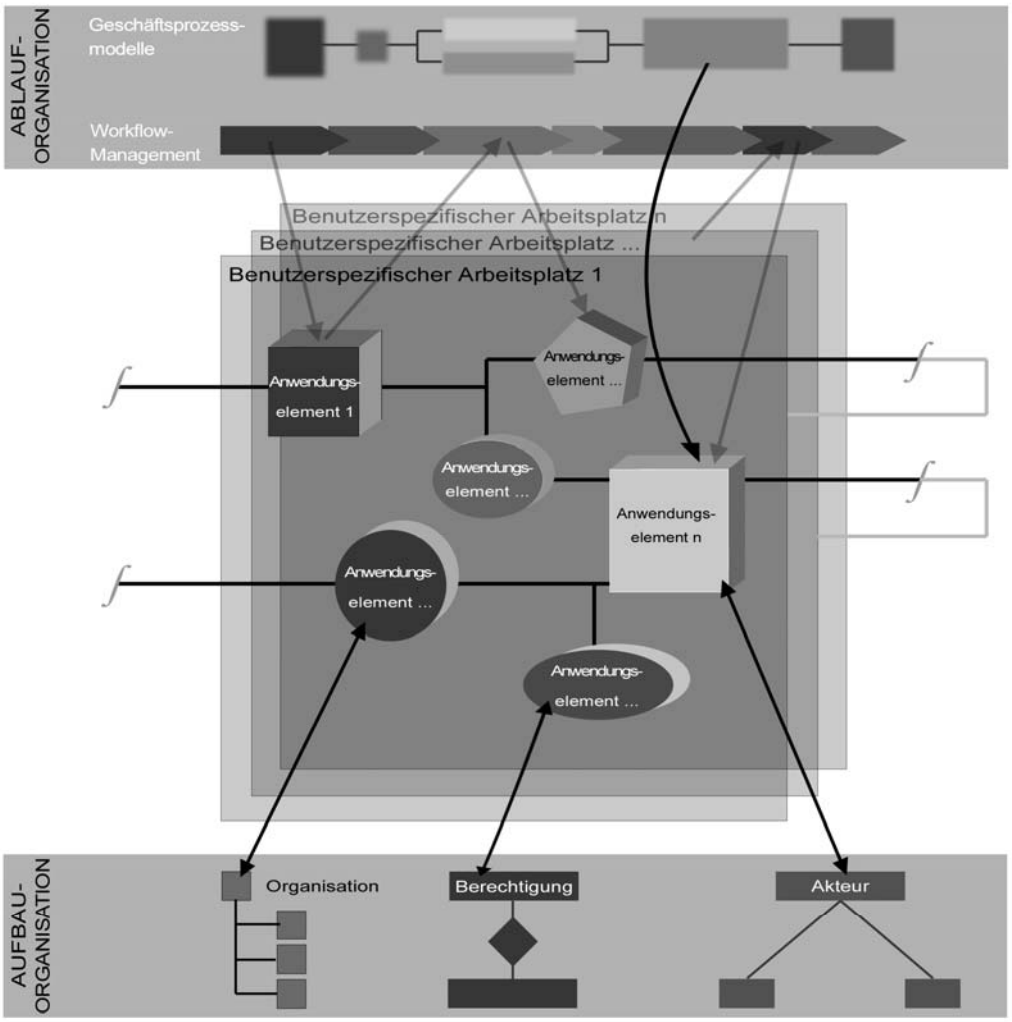


Abbildung 46: Benutzerspezifischer Arbeitsplatz

Vorgangsbearbeitung und Prozessmanagement identifizieren die Empfänger von Vorgangsbearbeitungs-Instanzen (Workitems) und den zugehörigen Anwendungs-elementen. Eine Zuordnung der Mitarbeiter zu ihren organisatorischen Einheiten ergänzt ihren Arbeitsplatz um weitere Anwendungselemente. Wird die organisatorische Zuordnung durch ein Berechtigungskonzept ergänzt, so können den Mitarbeitern dadurch weitere Anwendungselemente angeboten werden. Der Mitarbeiter als Akteur im System kann sich auch selbst Anwendungselemente aus einem für ihn freigegebenen Angebot zuordnen (Personalisierung).

Lösungsauswahl mit *TKMS*

Das Komponentenverwaltungssystem ***TKMS*** begleitet die Variantenkonfiguration, indem es die Ermittlung der Teilaufgaben durch die Gliederung des Anwendungsbereiches unterstützt und nach dem Festlegen der Teilaufgaben, eine Lösungssuche mittels Gegenstandsmustern der Anwendungselemente ermöglicht.

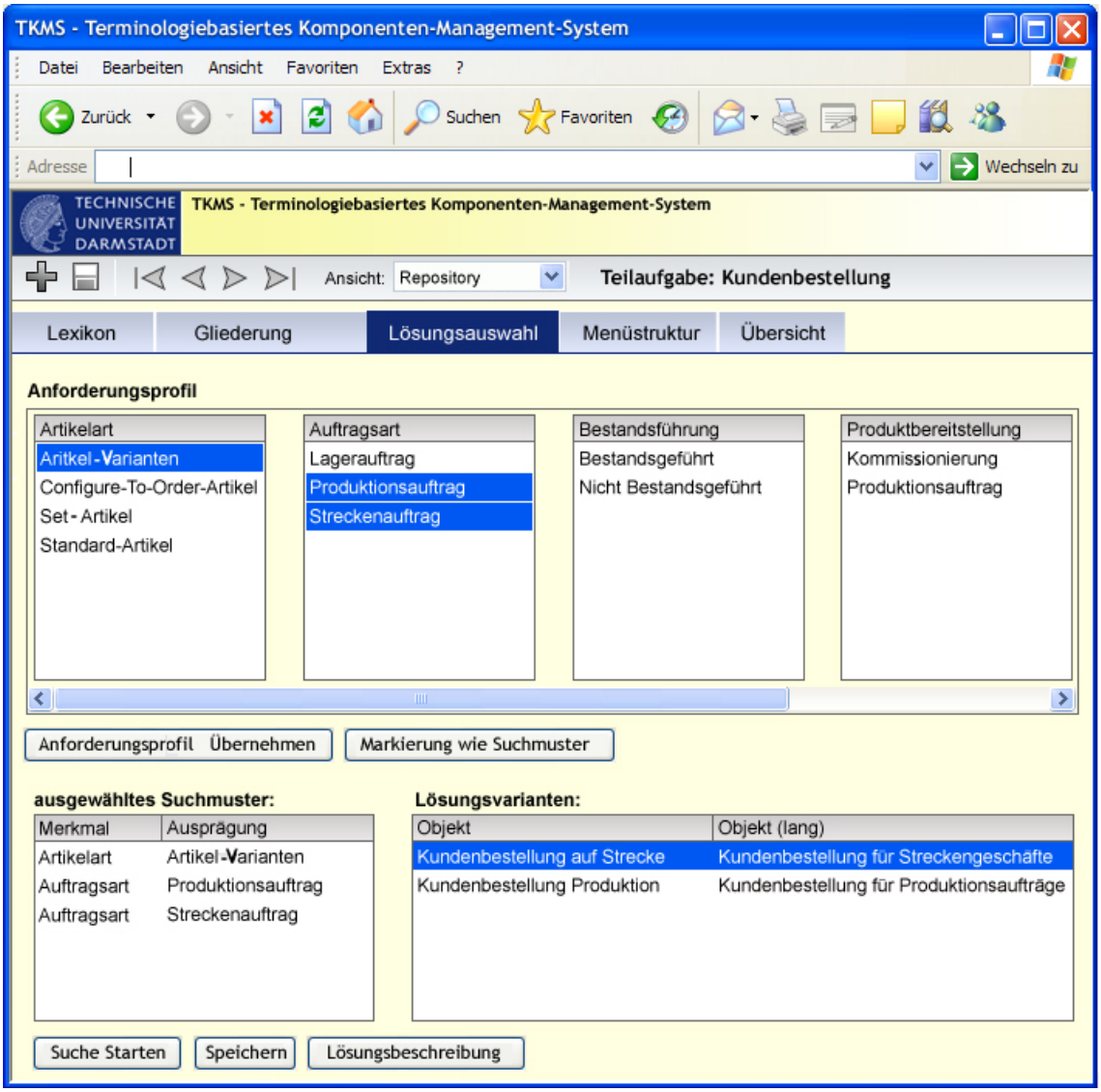


Abbildung 47: Lösungsauswahl im Anwendungselemente Repository

Eine zweite Ansicht „Repository“ des ***TKMS*** zeigt die Werkzeuge für Aufbau und Pflege des Lexikons und der Gliederung (Gliederung enthält Ordnungssystem und Basisverwaltungssystem) sowie das Konstruktionswerkzeug für die Lösungsauswahl. Über Anforderungsprofile werden passende Lösungen für Teilaufgaben gesucht. Aus der Ergebnismenge „Lösungsvarianten“ kann das gewünschte Anwendungselement

ausgewählt und der Gesamtlösung zugeführt werden. Nachdem alle Anwendungselemente ausgewählt sind, wird in einer Menüstruktur die Benutzerführung festgelegt und anschließend wird eine Gesamtanwendung, wie in Abschnitt 4.1.2 dargestellt, durch die Ausführung der Implementierungsvorschriften aller zugehörigen Anwendungselemente, implementiert.

4.3 Vereinfachte Variantenkonstruktion für Mittelstandsunternehmen

Für kleine und mittelständische Unternehmen sind die Verfahren der Variantenkonstruktion, die auf eine umfangreiche Anforderungsanalyse erfordern, zu aufwändig. Die ***Methode der semantischen Komposition*** ermöglicht es mit einem Partnerkonzept für homogene Branchen eine vorbereitete Variantenkonfiguration, basierend auf einem normierten Branchen-Aufgabenplan (siehe auch Abbildung 16 und Abbildung 38) durchzuführen. Der Partner erstellt eine branchentypische, fiktive Organisationsstruktur und ein für diese Branche passendes Prozessmodell. Darauf aufbauend wird ein Aufgabenplan, der auf den Erfahrungen von erfolgreich durchgeführten Projekten basiert, erstellt. Gemäß diesem Aufgabenplan werden die Lösungsvarianten aus dem Anwendungselemente-Baukastenssystem ausgewählt.

Fehlen Lösungsalternativen für genau diese Branche, so werden, wie bei einer Neukonstruktion, Entwicklungsergebnisse erstellt und in das Komponentenverwaltungssystem eingeführt. D.h. ein sprachbasiertes Fachkonzept, welches sich auf normierte Aussagen des Anwendungsbereiches abstützt, wird erstellt. Gegebenenfalls muss das Lexikon und die Gliederung erweitert werden. Idealerweise wird jedoch nur ein neues Anwendungselement hergestellt und mit seinem Gegenstandsmuster als Lösungsalternative in dem Komponentenverwaltungssystem abgelegt.

Existiert eine vorbereitete Variantenkonfiguration, dann kann das Kundenkonfigurations-Projekt viel effizienter durchgeführt werden. Thome und Hufgard haben Strategien zur optimierten Anforderungsnavigation entworfen [Thome/Hufgard96, 89ff]. Sie fordern eine regelbasierte interaktive Anforderungsnavigation. Sie schlagen ein Weglassen der IST-Analyse vor. Auf der Basis des vorbereiteten Lösungsangebots sollte eine erste „zufriedenstellende Eröffnungslösung“ erreichbar sein [Thome/Hufgard96, 93]. Thome und Hufgard haben eine Vorgehensweise zur Zerlegung des Lösungsangebotes nach Standard- und Optional-Lösungen (Standard/Optionsprinzip) mit der Reduktion auf das Notwendige und einer

Zuordnung typologischer (Teil-) Lösungen entwickelt, die für eine Projekteinführung von Standard-Anwendungssystemen, die mit der ***Methode der semantischen Komposition*** für Branchen vorgefertigt sind, gerade in Mittelstandsunternehmen einen durchgreifenden Erfolg verspricht.

Kapitel 5 Schlussbetrachtungen

Am Anfang jeder Forschung steht das Staunen.

Plötzlich fällt einem etwas auf.

Wolfgang Wickler[52]

Die Herstellung von Standard-Anwendungssystemen zeichnet sich durch ihre eigenen Probleme in der Entwicklung und ihre eigenen Faktoren für den Markterfolg aus. Es ist eine besondere Erfahrung für einen Anwendungsarchitekten oder einen Anwendungsentwickler, wenn die Zahl der Einführungsprojekte eines betriebswirtschaftlichen Anwendungssystems steigt. Damit nimmt auch die Vielfalt der Anforderungen an seine Programme zu. Mit wachsendem Erfolg einer Anwendungslösung mehren sich die Nachfragen der Kunden nach Funktionserweiterungen. Zudem

[52] Wolfgang Wickler (*1931), dt. Verhaltensforscher u. Zoologe

gehen Kunden davon aus, dass die Lösungen eines Herstellers in ihren funktionalen und prozessualen Abläufen und Datenstrukturen möglichst integriert sind. D.h. Kunden fordern eine schlanke und flexible Geschäftsabwicklung, möglichst ohne Redundanzen in der Datenerfassung. Alle Interaktions- und Ausgabemöglichkeiten sollten individuell an die organisatorischen und aufgabenorientierten Anforderungen ihres Unternehmens angepasst sein. Des Weiteren sollten Hersteller von Standard-Anwendungssystemen auf aktuelle Trends der Software- und Kommunikationstechnologie reagieren und das Beste dieser neuen Techniken mit den damit verbundenen neu erstellten bzw. verbesserten Anwendungslösungen ihren Kunden zur Verfügung stellen.

Außerdem erwarten Kunden, dass ihre einmal getätigten Investitionen für die Einführung ihres betriebswirtschaftlichen Anwendungssystems und die Anpassungen an ihr Unternehmen langfristigen Nutzen bringen. Das bedeutet für einen Hersteller von betriebswirtschaftlichen Anwendungssystemen, dass für neu entwickelte Konzepte und Lösungen immer auch ein Migrationspfad für Bestandskunden mitentwickelt werden muss. Gleichzeitig müssen sich die Hersteller von Standard-Anwendungssystemen auf eine verteilte, eventuell sogar globalisierte Entwicklung der Anwendungsprogramme einrichten und eine mehrstufige Fertigung von Kundenlösungen methodisch begünstigen. Gerade dieser permanente Wettlauf zwischen Entkopplung und Integration, sowohl für Technologie als auch für betriebswirtschaftliche Konzepte im Verbund mit ständig steigenden Anforderungen an Flexibilität und Beherrschbarkeit, stellen für die Systementwicklung besondere Herausforderungen dar.

In diesem Kontext betrachtet war es bisher fast unmöglich, eine aus Komponenten aufgebaute Anwendungslösung als Standard-Anwendungssystem erfolgreich im Markt zu positionieren. Versuche gab es schon viele (z.B. IBM San Francisco), dennoch ist kein wettbewerbsfähiges Produkt unter den Top 10 der betriebswirtschaftlichen Standard-Anwendungssysteme aus konfigurierbaren Komponenten erstellt worden. Die Idee, für eine komponentenorientierte Anwendungssystementwicklung eine Konstruktionsmethode aufbauend auf den Methoden und Verfahren traditioneller Ingenieurbereiche zu entwerfen, ist nicht neu [McIlroy69, Ott94, Klaeren94, Krause94].

Erfolg zeigt eine Konstruktionsmethode allerdings erst, wenn sie die drei folgenden Voraussetzungen erfüllt:

1. Methodische Erkenntnisse der technischen Konstruktionslehre werden konsequent auf die Konstruktion und Entwicklung von Standard-Anwendungssystemen angewandt. Das bedeutet, dass ein systematisches Vorgehen, ein Denken in Baugruppen, eine konsequente Wiederverwendung, eine Prozessstrukturierung sowie eine entsprechende Qualitätssicherung jedem Konstruktionsvorhaben für Anwendungssysteme zu Grunde liegen muss. Zudem muss die Konstruktionsmethode ganzheitlich durch ein Multipfad-Vorgehensmodell unterstützt werden.

2. Semantische Bauteile (Anwendungselemente) müssen den Komponentenbegriff in der Entwicklung, der Einführung und dem Betrieb dominieren. Das Architekturkonzept der Anwendungselemente ist das Schlüsselelement für eine erfolgreiche Variantenkonstruktion. Anwendungselemente, die normsprachlich definiert und eingebettet in ein anwendungsbereichsorientiertes Ordnungssystem sind, stellen den zentralen Bauplan für ein Baukastensystem für Anwendungssysteme dar.

3. Basierend auf einer serviceorientierten Architektur wird mit betriebswirtschaftlichen Grundfunktionen eine Business Services-Plattform entwickelt. Die Plattform stellt als Integrationsbasis die betriebswirtschaftliche Integration aller Kundenkonfigurationen sicher.

Die Erfüllung dieser drei Voraussetzungen kennzeichnet die ***Methode der semantischen Komposition***. Sie umfasst eine Vorgehensweise zur Herstellung, Einführung und Betreuung von Anwendungssystemen. Zudem werden die wesentlichen Einflussfaktoren auf die Systementwicklung und Konfiguration berücksichtigt: Ablauforganisation, Aufbauorganisation und Anwendungsbereich.

Konstruktionsprinzipien der technischen Konstruktionslehre für den Entwurf werden in der ***Methode der semantischen Komposition*** adaptiert und eingesetzt: Abstraktion - Konkretion und Komposition - Partition sowie die Entwurfsprinzipien Orthogonalisierung und Faktorenanalyse.

Ein Denken in Baugruppen wird methodisch kraft des puristischen Anspruchs einer Wiederverwendung durch Auswahl für die Variantenkonstruktion mit Baukastensystem eingefordert. Gerade diese klare Entscheidung ist es, die sich auf die Konzepte der ***Methode der semantischen Komposition*** erfolgreich auswirkt. Eine unbedingte Auswahlsystematik erfordert ein terminologiebasiertes Ordnungssystem mit Gliederung und Sachmerkmalleisten sowie semantische Relationen zwischen Anwendungselementen. Umgekehrt sind diese Konzepte, nach ihrer Erstellung, ausdrücklich geeignet für eine Variantenkonstruktion mit Lösungsauswahl.

Einige Ergebnisse der Forschungsarbeit sind hervorzuheben. Beispielsweise sind die fünf Regeln für eine Reduktion der Komplexität des Beziehungsnetzes zwischen Anwendungselementen oder der 5-Schritte-Plan für den Aufbau eines terminologiebasierten Ordnungssystems konkrete Handlungsanweisungen für die Umsetzung der ***Methode der semantischen Komposition*** in der Anwendungssystementwicklung.

Als besondere Auszeichnung der ***Methode der semantischen Komposition*** ist ihre Ganzheitlichkeit zu nennen. Die ***Methode der semantischen Komposition*** umfasst die Herausforderungen sowohl von Branchenentwicklungen als auch von Kundenkonfigurationen. Sie ist sowohl Teil des methodenneutralen Fachentwurfs als auch des Systementwurfs und der Komponentenkonfiguration. Sie betrachtet ebenso die Zusammenhänge zwischen Normspracherstellung und ihrer Verwaltung wie die Abhängigkeiten zwischen einem Ordnungssystem, einem Anwendungselemente-Baukastensystem und den betriebswirtschaftlich relevanten Filtersystemen. Zudem unterstützt sie methodisch die Anwendungskonfiguration unter Berücksichtigung des Einsatzes einer Integrationsbasis.

Diese ganzheitliche Betrachtung einer Anwendungssystementwicklung mit Variantenkonstruktion basierend auf Anwendungselementen war nur durch die Forschungsarbeit mit TKMS, einem Prototyp für ein terminologiebasiertes Komponenten-Management-System, und durch eine langjährige praktische Einsichtnahme in die Entwicklung von Standard-Anwendungssystemen möglich.

Nachdem die ***Methode der semantischen Komposition*** nun für ihren Einsatz in der Praxis zur Verfügung steht, gibt es für den Konstrukteur schon die nächste Herausforderung. Es gilt betriebswirtschaftliche Inhalte für den Entwurf der Anwendungselemente und den Aufbau des Ordnungssystems zu definieren und die

Konstruktionsanweisungen entsprechend der ***Methode der semantischen Komposition*** in die Unternehmenspraxis umzusetzen. Kein SOA-basiertes Standard-Anwendungssystem mit mehrstufiger Fertigung wird ohne solche Ordnungsstrukturen und ohne Unterstützung durch eine Konstruktionsmethode, wie hier beschrieben, einen erfolgreichen Weg in den globalen Markt der betriebswirtschaftlichen Anwendungssysteme finden. Sobald Branchenlösungen durch Partner des Herstellers und spezialisierte Lösungen in Kundenprojekten entstehen, kann, vom heutigen Stand her gesehen, nur ein konsequenter Einsatz der ***Methode der semantischen Komposition*** eine ingenieurgemäße Konstruktionsmethode sicherstellen.

Literaturverzeichnis

Aitken96	Aitken, Gary: Moving from C++ to Java. In: Dr. Dobb´s Journal, March 1996, S. 52-56.
Ashby74	Ashby, W. Ross: Einführung in die Kybernetik. Frankfurt: Suhrkamp, 1974.
Bachmann93	Bachmann, Reinhard: Gründe und Bedingungen des Erfolgs wissensbasierter Konfigurationssysteme. In: KI 1/1993, S. 48-50.
Batini et.al. 92	Batini, C.; S. Ceri; S.B. Navathe: Conceptual Database Design: An Entity-Relationship Approach. Redwood City: Benjamin Cummings 1992.
Becker et.al. 03	Becker, Steffen; Erich Ortner; Sven Overhage: Der Komponentenansatz – ein Riesenschritt für unsere geistige Entwicklung. In: Präsident der TU Darmstadt (Hrsg.): Thema Forschung – Vom Wort zum Bauelement. Monsheim: Verlag für Marketing und Kommunikation, 1/2003, S. 16-21.
Biggerstaff96	Biggerstaff, Ted: In: Panel: Component-Based Software Engineering (CBSE) in Proceedings: Fourth International Conference in Software Reuse, April 1996, Orlando, Florida, USA. Los Alamitos: IEEE Computer Society Press, 1996, S. 236-241.
Börding/Rahmer96	Abgleich komplexer Ressourcen beim ressourcenorientierten Konfigurieren. In: Sauer et.al. (Hrsg.): PuK-96, 10. Workshop „Planen und Konfigurieren“. St. Augustin: Infix . 15.-17. April 1996, S. 186 - 192.
BOMSIG94	Object Management Group: Business Object Management Special Interest Group (BOMSIG): Description of a Business Object, Version 2.0, 18. October 1994. Internet Download: http://www.omg.org/docs/1994/94-11-02.txt, August 1997.
Borowski61	Borowski, Karl-Heinz: Das Baukastensystem in der Technik. Berlin u.a.: Springer, 1961.
Börstler94	Börstler, Jürgen: Programmieren-im-Großen: Sprachen, Werkzeuge, Wiederverwendung. Technische Hochschule Aachen, Math.-Naturwiss. Fakultät, Dissertation, 1994. Umea Universitet, Sweden, ISBN 91-7174-959-4.

Booth et.al. 04 — Booth, David; Hugo Haas; Francis McCabe; Eric Newcomer; Michael Champion; Chris Ferris; David Orchard: Web Services Architecture: W3C Working Group Note 11 February 2004. Internet Download: http://www.w3.org/TR/2004/NOTE-ws-arch-20040211/, Oktober 2004.

Breiing/Flemming93 — Breiing, Alois; Manfred Flemming: Theorie und Methoden des Konstruierens. Berlin u.a.: Springer, 1993.

Brenner88 — Brenner, Walter: Entwurf betrieblicher Datenelemente: Ein Weg zur Integration von Informationssystemen. Berlin u.a.: Springer, 1988.

Bronstein et.al. 95 — Bronstein, I. N.; K. A. Semendjajew; G. Musiol; H. Mühlig: Taschenbuch der Mathematik. 2. Aufl. Frankfurt: Deutsch, 1995.

Broy/Siedersleben02 — Broy, Manfred; Johannes Siedersleben: Objektorientierte Programmierung und Softwareentwicklung: eine kritische Einschätzung. In: Informatik Spektrum 25, 2002, S. 3-11.

Bußmann90 — Bußmann, Hadumod: Lexikon der Sprachwissenschaft. 2. Auflage. Stuttgart: Alfred Kröner Verlag, 1990.

CCM02 — CORBA Component Model, June 2002. Internet Download: http://www.omg.org/docs/formal/02-06-65.pdf, September 2004.

Chen76 — Chen, P.P.: The Entitiy-Relationship Model, Towards an Unified View of Data. In: ACM Transactions on Database Systems, 1 (1976) 1, S. 9-36.

Chen93 — Chen, Sicheng: Retrieval of Reusable Components in a Deductive, Object-Oriented Database Environment. Technische Hochschule Aachen, Math.-Naturwiss. Fakultät, Dissertation, 1993.

Cherry91 — Cherry, George W.: System construction with object-oriented pictures. In: ACM Sigsoft, Software Engineering Notes, 16(4), 1991, S. 42-51. Zit. in: Prins, Robert: Developing business objects: a Framework driven approach. Maidenhead: McGraw-Hill, 1996, S. 96ff.

Coleman et.al. 94 — Coleman, Derek; Patrik Arnold; Stephanie Bodoff; Chris Dollin; Helena Gilchrist; Fiona Hayes; Paul Jeremeas: Object-Oriented Development: The Fusion Method. Englewood Cliffs: Prentice Hall, 1994.

Convent/Wernecke94 — Convent, Bernhard; Wolfgang Wernecke: Bausteinverwaltung und Suchunterstützung – Basis für die Software-Wiederverwendung. In: Theorie und Praxis der Wirtschaftsinformatik, Heft 180, 31. Jg., November 1994, S. 59-69.

Cooper99	Cooper, Alan: The Inmates Are Running the Asylum. Indianapolis: SAMS, 1999.
Curtis04	Curtis, Dave: RMI, IIOP, and EJB. Internet Download: http://info.borland.com/devsupport/appserver/faq/dave-curtis-rmiiiop.html, September 2004.
Dahme/Raeithel97	Dahme, Christian; Arne Raeithel: Ein tätigkeitstheoretischer Ansatz zur Entwicklung von brauchbarer Software. In: Informatik Spektrum 20, 1997, S.5-12.
DeRemer/Kron76	DeRemer, Frank; Hans H. Kron: Programming-in-the-Large Versus Programming-in-the-Small. In: IEEE Transactions in Software Engineering, Vol SE-2, No. 2, June 1976, S. 80-86.
DIN 4000-92	DIN Norm. DIN 4000 Teil 1: Sachmerkmal-Leisten: Begriffe und Grundsätze, September 1992. Deutsches Institut für Normung e.V.
DIN 4000-93	DIN Norm. DIN 4000 Teil 2 bis 79: Sachmerkmal-Leisten für Norm- und Konstruktionsteile. Deutsches Institut für Normung e.V. Zitiert nach Pahl/Beitz93, 491.
DIN Variantenübersicht86	DIN-Manuskriptdruck: Variantenübersicht: Variante, Variantenstückliste. Leitfaden zum Erstellen eines maschinellen Varianten-Suchsystems mit automatischer Variantenstücklisten-Auswahl für alle Bereiche eines Unternehmens. Berlin: Beuth, 1986.
Dörner87	Dörner, Dietrich: Problemlösen als Informationsverarbeitung. 3. Aufl. Stuttgart u.a.: Kohlhammer, 1987.
Dreibholz75	Dreibholz, D.: Ordnungsschemata bei der Suche nach Lösungen. In: Konstruktion 27 (1975), S. 233-239.
Duden96	Dudenverlag: LexiRom Version 2.0. Microsoft Corporation und Bibliographisches Institut & F.A. Brockhausverlag AG, 1996.
Duden04	Duden-Suche auf der Duden Homepage: http://www.duden.de/, Oktober 2004.
ebXML04	ebXML: Core Components Standardization at http://ebxml.org./. Internet Download: http://ebxml.org./specs/ccDICT.pdf, Oktober 2004.
Eckes91	Eckes, Thomas: Psychologie der Begriffe: Strukturen des Wissens und Prozesse der Kategorisierung. Göttingen u. a.: Hogrefe, 1991.
Eisenecker95	Eisenecker, Ulrich W.: Der Weg zur flinken Software. In: i/X 9/1995, S. 164 – 169.
Fähnrich et.al. 97	Fähnrich, K.-P.; I. Schöbe; J. Kunsmann; A. Gräble: Componentware. In: Offene Systeme (1997) 6: 140-150.

Ferstl/Sinz94	Ferstl, Otto K.; Elmar J. Sinz: Der Ansatz des semantischen Objektmodells (SOM) zur Modellierung von Geschäftsprozessen, 1994. Internet Download: http://www.iaws.wiai.uni-bamberg.de/veroeffentlichungen/bam_beitraege/no21.pdf , 1996.
Fitié95	Fitié, Bert: Component Software and Oberon: a Perspective on Oberon/F. In: The Oberon Tribune, No 1/1, July 1995, S. 3.
Fowler/Scott97	Fowler, Martin; Kedall Scott: UML Distilled: Applying the Standard Object Modeling Language. Reading, Massachusetts: Addison-Wesley, 1997.
Goldberg/Rubin95	Goldberg, Adele; Kenneth S. Rubin: Succeeding with Objects: Decision Framework for Project Management. Addison-Wesley, 1995.
Grabowski et.al. 93	Grabowski, H; St. Rude; A. Suhm; G. Staub: Lösungsmusterbasierte Produktmodellierung in wissenbasierten Konstruktionssystemen. In: VDI-Berichte 1079: Rechnerunterstützte Wissenverarbeitung in Entwicklung und Konstruktion `93. Tagung Heidelberg, Sept. 1993. VDI-Gesellschaft Entwicklung, Konstruktion, Vertrieb. Düsseldorf: VDI-Verlag, 1993, S. 55 – 80.
Graf97	Graf, Peter: Komponenten in betriebswirtschaftlicher Standardsoftware: Das Business Framework der SAP. In HMD – Handbuch der modernen Datenverarbeitung, 197/1997, S. 62-75.
Greenfield/Short04	Greenfield, Jack; Keith Short: Moving to Software Factories. July 2004. Internet Download: http://www.softwarefactories.com/ScreenShots/MS-WP-04.pdf, September 2004.
Grupp95	Grupp, Bruno: Aufbau einer optimalen Stücklistenorganisation: offene Stücklisten, Variantengenerator, PPS-Rahmen, CAD-Connection, Praxisbeispiele. Rennigen-Malmsheim: expert-Verlag, 1995.
Gülden/Günter95	Gülden, Oliver; Andreas Günter: Konzept für die Realisierung einer ressourcenorientierten Vorgehensweise in KONWERK. In: Günter, Andreas (Hrsg.): Wissensbasiertes Konfigurieren: Ergebnisse aus dem Projekt PROKON. St. Augustin: Infix, 1995, S. 237 – 244.
Günter et.al. 94	Günter, Andreas; Lothar Hotz; Thomas Vietze: Diskussion über die Verwendung von terminologischen Systemen oder modularen Problemlösungsbausteinen für die Wissensrepräsentation in Konfigurierungssystemen. In: Bergmann et. al. (Hrsg.): PuK-94, 8. Workshop „Planen und Konfigurieren“. SEKI Working Paper SWP-94-01. 18.-19. April 1994, S. 59 - 67.

Günter93	Günter, Andreas: Verfahren zur Auflösung von Konfigurationskonflikten in Expertensystemen. In: KI 1/1993, S. 16 – 23.
Günter95	Günter, Andreas (Hrsg.): Wissensbasiertes Konfigurieren: Ergebnisse aus dem Projekt PROKON. St. Augustin: Infix, 1995.
Günter95b	Günter, Andreas: Das Projekt PROKON im Überblick. In: Günter, Andreas (Hrsg.): Wissensbasiertes Konfigurieren: Ergebnisse aus dem Projekt PROKON. St. Augustin: Infix, 1995, S. 3 - 10.
Günter95c	Günter, Andreas: Architektur des domänenunabhängigen Konfigurierungswerkzeuges KONWERK. In: Günter, Andreas (Hrsg.): Wissensbasiertes Konfigurieren: Ergebnisse aus dem Projekt PROKON. St. Augustin: Infix, 1995, S. 49 – 60.
Hansen et.al. 92	Hansen, Hans Robert; Robert Mühlbacher; Gustaf Neumann: Begriffsbasierte Integration von Systemanalysemethoden. Heidelberg: Physica-Verlag, 1992.
He03	He, Hao: What is Service-Oriented Architecture? Veröffentlicht September 2003. Internet Download: http://webservices.xml.com/pub/a/ws/2003/09/30/soa.html, Oktober 2004.
Heinrich93	Heinrich, Michael: Ressourcenorientiertes Konfigurieren. In: KI 1/1993, S. 11 – 15.
Henninger97	Henninger, Scott: An Evolutionary Approach to Constructing Effective Software Reuse Repositories. In: ACM Transactions on Software Engineering and Methodology, Vol. 6, No. 2, April 1997, S. 111-140.
Heß/Scheer92	Heß, Helge; August-Wilhelm Scheer: Retrieval wiederverwendbarer Softwarebausteine. In: Wirtschaftsinformatik, 34. Jg., Heft 2, April 1992, S. 190-200.
Hobbs94	Hobbs, Elizabeth T.: A Uniform Data Model for Reuse Library Interoperability. Internet-Download: ftp://ftp.umcs.maine.edu/pub/wisr6/proceedings/ps/hobbs.ps, Dezember 1994.
Hofstätter66	Hofstätter, P.R.: Psychologie. Frankfurt/M.: Fischer Bücherei, 1966.
Holliger82	Holliger-Uebersax, Hermann: Morphologie. Bericht. Morphologisches Institut Zürich, 1982.

Horstmann/Kirtland97 Horstmann, Markus; Mary Kirtland: DCOM Architecture. Microsoft Developer Network, veröffentlicht 1997. Internet Download: http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dndcom/html/msdn_dcomarch.asp, Oktober 2004.

Hruschka86 Hruschka, Peter: Prinzipien der Modularisierung. In HMD – Handbuch der modernen Datenverarbeitung, 130/1986, S. 67-75.

Hüber et.al. 03 Hüber, Rainer; Klaus-Peter Lang; Wolfgang Weiss; Thomas Reiss; Rainer Zinow: Customer Fitting Terminology – Conceptual ideas for a terminology based development process. Strategischer Konzeptentwurf der SAP AG – nur zur internen Verwendung, Dez. 2003.

Hufgard et.al. 05 Hufgard, Andreas; Heiko Hecht; Wolfgang Walz; Frank Hennermann; Gerald Brosch; Sabine Mehlich; Christian Bätz: Business Integration mit SAP-Lösungen. Berlin: Springer, 2005.

Jablonski et.al. 97 Jablonksi, Stefan; Markus Böhm; Wolfgang Schulze (Hrsg.): Workflow-Management: Entwicklung von Anwendungen und Systemen; Facetten einer neuen Technologie. Heidelberg: dpunkt-Verl., 1997.

Kalkmann et.al. 96 Kalkmann, Jörg; Klaus-Peter Lang; Erich Ortner: Anwendungselemente. Bericht 78-96, Universität Konstanz Informationswissenschaft, April 1996.

Keller/Popp96 Keller, Gerhard; Karl Popp: Neue Epoche der Software-konfiguration in SAPinfo: Continuous Business Engineering, 1996, S. 12-18.

Kirtland97 Kirtland, Mary: The COM+ Programming Model Makes it Easy to Write Components in Any Language. In: Microsoft Systems Journal, Dezember 1997. Internet Download: http://www.microsoft.com/msj/1297/complus2/complus2.aspx, September 2004.

Klaeren94 Klaeren, H.: Probleme des Software-Engineering: Die Programmiersprache – Werkzeug des Softwareentwicklers. In: Informatik Spektrum 17, 1994, S. 21-28.

Klaus71 Klaus, Georg (Hrsg.): Wörterbuch der Kybernetik. Band I: Abbildtheorie – Meßwertwandler. Frankfurt/Main: Fischer Bücherei, 1971.

Koller85 Koller, Rudolf: Konstruktionslehre für den Maschinenbau: Grundlagen des methodischen Konstruierens. 2. Aufl. Berlin u.a.: Springer, 1985.

Koller/Kastrup94 Koller, Rudolf; Norbert Kastrup: Prinziplösungen zur Konstruktion technischer Produkte. Berlin: Springer, 1994.

Kotler89 Kotler, Philip: Marketing-Management: Analyse, Planung und Kontrolle. 4. Aufl. Stuttgart: Poeschel, 1989.

Krause93 Krause, Werner (Hrsg.): Konstruktionselemente der Feinwerktechnik. 2. Aufl. München, Wien: Hanser, 1993.

Krause94 Krause, Gerrit: Neue Software braucht die Welt. In: Business Computing 12, 1994, S. 72-75.

Krauser86 Krauser, Dieter: Methodik zur Merkmalbeschreibung technischer Gegenstände. Hrsg. DIN, Deutsches Institut für Normung e.V. Berlin, Köln: Beuth Verlag, 1986.

Lang/Kalkmann97 Lang, Klaus-Peter; Jörg Kalkmann: Wirtschaftliche Vorteile von komponentenorientierten Informationssystemen. In: Reiterer, Harald; Thomas Mann (Hrsg.): Informationssysteme als Schlüssel zur Unternehmensführung – Anspruch und Wirklichkeit. Proceedings des 3. Konstanzer Informationswissenschaftlichen Kolloquiums (KIK ,97). Konstanz: UVK, Univ.-Verl. Konstanz, 1997.

Lang97 Lang, Klaus-Peter: Komponentenorientierte Entwicklung und Konfiguration betriebswirtschaftlicher Standardsoftware. Konzeptpapier für ein Kooperationsprojekt der TU Darmstadt mit der SAP AG, Dez. 1997.

Lang98 Lang, Klaus-Peter: Variantenkonstruktion betriebswirtschaftlicher Anwendungssoftware, Teil1: Methode der semantischen Komposition. Arbeitsbericht 98/02 des Fachgebiets Wirtschaftsinformatik I, Entwicklung von Anwendungssystemen der Technischen Universität Darmstadt, Hrsg.: Prof. Dr. Erich Ortner, 1998.

Lehmann/Ortner97 Lehmann, Frank; Erich Ortner: Entwicklung von Workflow-Management-Anwendungen im Kontext von Geschäftsprozeß- und Organisationsmodellierung. In: Information Management 4/97, S. 62-69.

Lehmann99 Lehmann, Frank R.: Fachlicher Entwurf von Workflow-Management-Anwendungen. Stuttgart, Leipzig: Teubner, 1999.

Loos99 Loos, Peter: Grunddatenverwaltung und Betriebsdatenerfassung als Basis der Produktionsplanung und –steuerung. In Corsten, H.; Friedl, B. (Hrsg.): Produktionscontrolling, (Verlag Vahlen) München 1999, S. 227-252.

Lorenzen74 Lorenzen, Paul: Konstruktive Wissenschaftstheorie. Frankfurt/M: Suhrkamp Taschenbuch Wissenschaft 93, 1974.

Martin89 Martin, James: Information Engineering, Book I – Introduction. Bd. 1 von 3. Englewood Cliffs: Prentice Hall, 1989.

Martin/Odell95	Martin, James; James J. Odell: Object-oriented methods: a foundation. Englewood Cliffs: Prentice Hall, 1995.
Mayer79	Mayer, Richard E.: Denken und Problemlösen: Eine Einführung in menschliches Denken und Lernen. Berlin u.a.: Springer, 1979.
McIlroy69	Mc Ilroy, M.D.: Mass produced Software Components. In: Naur, Peter; Brian Randell (Hrsg.): Software Engineering: Report on a conference, Garmisch, Oktober 1968. NATO SCIENCE COMMITTEE, 1969.
Meinl90	Meinl, Franz: Sachmerkmale: Schlüssel zur technischen Gestaltung, Beschreibung und Information. Ehningen bei Böblingen: expert-Verlag, 1990.
Mertens et.al. 95	Mertens, Peter; Jochen Holzner; Petra Ludwig: Branchensoftware. In: Informatik Spektrum 18, 1995, S. 340-341.
Meyer96	Meyer, Bertrand: Erfolgsschlüssel Objekttechnologie: Managerführer zur Neuorganisation des Softwareprozesses. München: Hanser, 1996.
Mezini et.al. 03	Mezini, Mira; Michael Eichberg; Michael Haupt: Komponentenbasierte Softwaresysteme: historische Abriss. In: Präsident der TU Darmstadt (Hrsg.): Thema Forschung – Vom Wort zum Bauelement. Monsheim: Verlag für Marketing und Kommunikation, 1/2003, S. 130-134.
Microsoft02	Microsoft: Application Architecture for .Net: Designing Applications and Services. Microsoft Corporation, 2002, (ISBN 0-7356-1837-2).
Müller90	Müller, Johannes: Arbeitsmethoden der Technikwissenschaften: Systematik, Heuristik, Kreativität. Berlin u.a.: Springer, 1990.
Nasvytis53	Nasvytis, A.: Die Gesetzmäßigkeiten kombinatorischer Technik. Berlin u.a.: Springer, 1953.
Nebel95	Nebel, Bernhard: Komplexitätsanalysen in der Künstlichen Intelligenz. In: KI 2/1995, S. 6-14.
Ning et.al. 94	Ning, Jim Q., Kanth Miriyala; W. Kozaczynski: An Architecture-driven, Business-specific and Component-based Approach to Software Engineering. In: W.B. Frakes (Hrsg.): Proceedings of the Third International Conference on Software Reuse. IEEE Computer Society Press, 1994, S. 84-93.
Norelem96	Katalog der Firma Norelem Normelemente: Flexibles Normteilesystem für wirtschaftliche Konstruktionen. Markgröningen, Dezember 1996.

Orfali/Harkey95	Orfali, Robert; Dan Harkey: Object Component Suites: The Whole is Greater than the Parts. In: Datamation, February 15, 1995, S. 44-47.
Orfali et.al. 96	Orfali, Robert; Dan Harkey; Jeri Edwards: The Essential Distributed Objects Survival Guide. New York: John Wiley & Sons, 1996.
Ortner/Söllner89	Ortner, Erich; B. Söllner: Semantische Datenmodellierung nach der Objekttypenmethode. In: Informatik Spektrum 12, 1989, S. 31-42.
Ortner83	Ortner, Erich: Aspekte einer Konstruktionssprache für den Datenbankentwurf. Darmstadt: S. Toeche-Mittler Verlag, 1983.
Ortner95	Ortner, Erich: Abstraktion und Komposition. Universität Konstanz, Informationswissenschaft, Bericht 66-95, 1995.
Ortner97	Ortner, Erich: Methodenneutraler Fachentwurf: Zu den Grundlagen einer anwendungsorientierten Informatik. Stuttgart: Teubner, 1997.
Ortner97b	Ortner, Erich: Entwicklung von Anwendungssystemen II: Modellierungsmethoden. Vorlesungsskriptum, TU Darmstadt, FG Wirtschaftsinformatik I, Institiut für Betriebswirtschafts-lehre, 1997.
Ortner et.al. 99	Ortner, Erich; Klaus-Peter Lang; Jörg Kalkmann: Ein Szenario für die Anwendungssystementwicklung mit fachlichen Komponenten. In: Bernd Britzelmaier; Stephan Gerberl (Hrsg.): Wirtschaftsinformatik als Mittler zwischen Technik, Ökonomie und Gesellschaft. Stuttgart: Teubner, 1999.
Ortner et.al. 99b	Ortner, Erich; Klaus-Peter Lang; Jörg Kalkmann: Anwendungssystementwicklung mit Komponenten. In: IM Information Management & Consulting 14 (1999) 2, S. 35-45.
Ortner05	Ortner, Erich: Sprachbasierte Informatik: Wie man mit Wörtern die Cyber-Welt bewegt. Noch zu erscheinen - geplant 2005.
Osgood et.al. 78	Osgood, Ch. E.; J.S. Suci; P. H. Tannenbaum: The Measurement of Meaning. 4th Printing, University of Illinois, 1978.
Osgood73	Osgood, Charles E.:Eine Entdeckungsreise in die Welt der Begriffe und Bedeutungen. In: Schramm, Wilbur (Hrsg.): Grundfragen der Kommunikationsforschung. 5. Aufl. München: Juventa, 1973, S. 39-54.
Ostermann04	Ostermann, Klaus: Bessere Software durch Querschneidende Module. Ausgezeichnete Informatikdissertationen 2003, GI-Edition Lecture Notes in Informatics, 2004.

Ostertag et.al. 92	Ostertag, E.; J. Hendler; R. Prieto-Diáz; CH. Braun: Computing Similarity in a Reuse Library System: An AI-Based Approach. In: ACM Transactions on Software Engineering and Methodology, Vol. 1, No. 3, July 1992, S. 205-228.
Ott94	Ott, Hans Jürgen: Das „ingenieurgemäße“ am Software Engineering. In: GI Softwaretechnik-Trends: Mitteilungen der Fachgruppen ‘Software-Engineering’ und ‘Requirements-Engineering’, Band 14, Heft 1, Februar 1994.
Pahl/Beitz93	Pahl, Gerhard; Wolfgang Beitz: Konstruktionslehre: Methoden und Anwendung. 3. Aufl. Berlin u.a.: Springer, 1993.
Pfitzner93	Pfitzner, Kai: Fallbasierte Konfigurierung. In: KI 1/1993, S. 24 – 30.
Pintado95	Pintado, Xavier: Gluons and the Cooperation between Software Components. In: Oscar Nierstrasz, Dennis Tsichritzis (Hrsg.): Object-Oriented Software Composition. London u.a.: Prentice Hall, 1995.
Poulin/Werkman96	Poulin, Jeffrey S.; Keith J. Werkman: Merlding Structured Abstracts and the World Wide Web for Retrieval of Reusable Components. Internet-Download: http://pooh.unl.edu/~scotth/ssr95/poulin/poulin.html, Februar 1996.
Pree97	Pree, Wolfgang: Komponentenbasierte Softwareentwicklung mit Frameworks. Heidelberg: dpunkt, 1997.
Pree/Koskimies99	Pree Wolfgang; Kai Koskimies: Framelets – Small is Beautiful. In: Mohamed E. Fayad et.al. (Hrsg.): Building Application Frameworks: Object-Oriented Foundations of Framework Design. New York: John Wiley&Sons, 1999, S. 411-413.
Prieto-Diáz/Freeman87	Prieto-Diáz, Rubén; Peter Freeman: Classifying Software for Reusability. In: IEEE Software, January 1987, S. 6-16.
Prieto-Diáz/ Neighbors86	Prieto-Diáz, Rubén; James M. Neighbors: Module Interconnection Languages. In: The Journal of Systems and Software 6, (1986), S. 307-334.
Prieto-Diáz93	Prieto-Diáz, Rubén: Status Report: Software Reusability. In: IEEE Software, May 1993, S. 61-66.
Prins96	Prins, Robert: Developing business objects: a Framework driven approach. Maidenhead: McGraw-Hill, 1996.
Puppe91	Puppe, Frank: Eingführung in Expertensysteme. 2. Aufl. Berlin u.a.: Springer, 1991.
Reimer91	Reimer, Ulrich: Einführung in die Wissenrepräsentation: netzartige und schema-basierte Repräsentationsformate. Stuttgart: Teubner, 1991.

REX96	REX – Reconfigurable & Extensible Parallel and Distributed Systems. ESPRIT Projekt 2080. Ergebnisveröffentlichung 1996.
Richter84	Richter, W.: Gliedern eines Projekts. In: Konstruktion 36 (1984) H. 12, S. 488-490.
Roth94 Bd. 1	Roth, Karlheinz: Konstruieren mit Konstruktionskatalogen. Band. 1: Konstruktionslehre. 2.Aufl. Berlin u.a.: Springer, 1994.
Roth94 Bd. 2	Roth, Karlheinz: Konstruieren mit Konstruktionskatalogen. Band. 2: Kataloge. 2.Aufl. Berlin u.a.: Springer, 1994.
Schädler95	Schädler, Kristina: Ansätze zur Abhängigkeitsverwaltung für KONWERK. In: Günter, Andreas (Hrsg.): Wissensbasiertes Konfigurieren: Ergebnisse aus dem Projekt PROKON. St. Augustin: Infix, 1995, S. 217 – 228.
Scheer94	Scheer, August-Wilhelm: Wirtschaftsinformatik: Referenzmodelle für industrielle Geschäftsprozesse. 4. Aufl., Berlin u.a.: Springer, 1994.
Schienmann94	Schienmann, Bruno: Die Teil/Ganze-Beziehung im objektorientierten Fachentwurf. In: Lipeck, U.W.; Vossen, G. (Hrsg.): formale Grundlagen für den Entwurf von Informationssystemen. Proc. GI-Workshop 1994, Tutzing. Informatik-Berichte 03/94. Hannover: Universität, 1994, S. 43-59.
Schienmann97	Schienmann, Bruno: Objektorientierter Fachentwurf: ein terminologiebasierter Ansatz für die Konstruktion von Anwendungssystemen. Stuttgart: Teubner, 1997.
Schmid et.al. 95	Schmid, Hans A.; Clemens Ballarin; Francesco Indolfo: Konstruktion eines Geschäftsprozeß-Baukastens zur Steuerung von Fertigungszellen. In: Objektspektrum 5/95, S. 42-49.
Schmitz-Esser95	Schmitz-Esser, Winfried: Ein Thesaurus als Teil eines terminologischen Lexikons. In: Meder et. al. (Hrsg.): Konstruktion und Retrieval von Wissen, 3. Tagung der deutschen ISKO Sektion, vom 27. – 29.10.1993. Frankfurt/M: INDEKS Verlag, 1995, S. 47 – 54.
Scholz/Volmering04	Scholz, Thorsten; Thomas Volmering: Business Process Management with SAP NetWeaver: Modelling, Executing, and Monitoring Business Processes. SAP Community: SAPPHIRE04, New Orleans, 2004. Internet Download: https://www.sdn.sap.com/sdn/index.sdn, September 2004.
Shaw/Garlan96	Shaw, Mary; David Garlan: Software Architecture: Perspective on an Emerging Discipline. Upper Saddle River: Prentice Hall, 1996.

Sinz96	Sinz, Elmar J.: Ansätze zur fachlichen Modellierung betrieblicher Informationssysteme: Entwicklung, aktueller Stand und Trends. In: Heilmann, Heinrich und Roithmayr: Information Engineering. München, Wien: Oldenbourg Verlag, 1996, S. 123 – 143.
Steinhilper/Röper94	Steinhilper, Waldemar; R. Röper: Maschinen und Konstruktionselemente. 4. Aufl., Berlin u.a.: Springer, 1994.
Stoer89	Stoer, Josef: Numerische Mathematik 1. 5. Aufl. Berlin u.a.: Springer, 1989.
Sullo94	Sullo, Gary C.: Object Engineering: designing large scale, object-oriented systems. New York: John Wiley & Sons, 1994.
Taligent94	Taligent Glossary (1994). Heute IBM OT Library. www.software.ibm.com.
Tank93	Tank, Wolfgang: Wissensbasiertes Konfigurieren: Ein Überblick. In: KI 1/1993, S. 7 – 10.
Taylor95	Taylor, David A.: Business Engineering with Object Technology. New York: John Wiley, 1995.
Thome/Hufgard96	Thome, Rainer; Andreas Hufgard: Continuous System Engineering: Entdeckung der Standardsoftware als Organisator. Würzburg: Vogel, 1996.
Turowski01	Turowski, Klaus: Spezifikation und Standardisierung von Fachkomponenten. In: Wirtschaftsinformatik 43, 2001, 3, S. 269-281
Turowski02	Turowski, Klaus (Hrsg.): Vereinheitlichte Spezifikation von Fachkomponenten – Memorandum des GI Arbeitskreises 5.10.3 Komponentenorientierte betriebliche Anwendungs-systeme. Universität Augsburg, 2002. Internet Download: http://www.fachkomponenten.de/, August 2004.
UML04	Unified Modelling Language Standardization at http://www.UML.org. Internet Download von Rational: http://www3.software.ibm.com/ibmdl/pub/software/rational/web/whitepapers/2003/Tp180.PDF, September 2004.
VDI 2212-81	VDI Richtlinie VDI 2212: Systematisches Suchen und Optimieren konstruktiver Lösungen, Oktober 1981. Verein Deutscher Ingenieure.
VDI 2215-80	VDI Richtlinie VDI 2215: Organisatorische Voraussetzungen und Hilfsmittel, Novemer 1980. Verein Deutscher Ingenieure.
VDI 2221-93	VDI Richtlinie VDI 2221: Methodik zum Entwickeln und Konstruieren technischer Systeme und Produkte, Mai 1993. Verein Deutscher Ingenieure.

VDI 2222-97	VDI Richtlinie VDI 2222 Blatt 1: Konstruktionsmethodik: Methodisches Entwickeln von Lösungsprinzipien, Juni 1997. Verein Deutscher Ingenieure.
VDI 2222-82	VDI Richtlinie VDI 2222 Blatt 2: Konstruktionsmethodik: Erstellung und Anwendung von Konstruktionskatalogen, Februar 1982. Verein Deutscher Ingenieure.
VDI-Berichte93	VDI-Berichte 1079: Rechnerunterstützte Wissenverarbeitung in Entwicklung und Konstruktion `93. Tagung Heidelberg, Sept. 1993. VDI-Gesellschaft Entwicklung, Konstruktion, Vertrieb. Düsseldorf: VDI-Verlag, 1993.
Wedekind/Müller81	Wedekind, Hartmut; Theo Müller: Stücklistenorganisation bei einer großen Variantenzahl. In: Angewandte Informatik 23, 9, 1981, S. 377-383.
Wedekind/Ortner80	Wedekind, Hartmut; Erich Ortner: Systematisches Konstruieren von Datenbankanwendungen: Zur Methodologie der angewandten Informatik. München: Hanser, 1980.
Wedekind81	Wedekind, Hartmut: Datenbanksysteme I: Eine konstruktive Einführung in die Datenverarbeitung in Wirtschaft und Verwaltung. 2. Aufl. Mannheim u.a.: Bibliographisches Institut, 1981.
Wedekind92	Wedekind, Hartmut: Objektorientierte Schemaentwicklung: ein kategorialer Ansatz für Datenbanken und Programmierung. Mannheim u.a.: BI-Wiss.-Verl., 1992.
Weide et.al. 91	Weide, Bruce W.; William F. Ogden; Stuart H. Zweben: Reusable Software Components. In: Yovits, Marshall C. (Hrsg.): Advances in Computers, Vol. 33. Boston u.a.: Academic Press, 1991, S. 1-65.
Woods03	Woods, Dan: Enterprise Service Architecture. Sebastopol, USA: O'Reilly, 2003.
Zeller/Snelting97	Zeller, Andreas; Gregor Snelting: Unified Versioning throug Feature Logic. Universität Braunschweig, Software-technologie, Informatik-Bericht No. 96-01, 1997.
Zeller97	Zeller, Andreas: Configuration Management with Version Sets: A Unified Software Versioning Model and it`s Applications. Universität Braunschweig, Mathematik und Informatik, Dissertation, 1997. http://www.cs.tu-bs.de/softech/papers/zeller-phd/
Zencke04	Zencke, Peter: The Five Key Elements of Enterprise Services Architecture: Next Business Solutions on the SAP NetWeaver Enterprise Services Platform. Keynote auf der SAP TechEd in San Diego, Oktober 2004. Internet Download: https://www.sdn.sap.com/sdn/index.sdn , Oktober 2004.

Zimmermann93 Zimmermann, Martin: Konstruktion und Management verteilter Anwendungen. Dissertation. Wiesbaden: Deutscher Universitätsverlag, 1993.

Zwicky71 Zwicky, Fritz: Entdecken, Erfinden, Forschen im Morphologischen Weltbild. München: Droemer Knaur, 1971.

Lebenslauf

Vorname:	Klaus-Peter
Name:	Lang
Adresse:	Dahlienstrasse 13 74336 Brackenheim-Botenheim
Telefonnummer:	07135/961924
E-mail:	klaus-peter.lang@gmx.de
Geburtsdatum:	19. Januar 1963
Nationalität:	deutsch
Familienstand:	verheiratet, 3 Kinder

Wissenschaftliche Ausbildung:

1997 - 1998	Forschungsjahr (18 Monate) Teilnahme am Projekt „Terminologiebasiertes Komponenten-Management-System (***TKMS***)“. ***TKMS*** ist ein Forschungs-Prototyp, der in Kooperation mit der Bereich Wirtschafts-informatik I, Entwicklung von Anwendungssystemen, der TU Darmstadt konzipiert und implementiert wurde.
1993 – 1995	Universität Konstanz **Diplom Informationswissenschaftler** Schwerpunkte: Informationsmanagement und Software Engineering
1988 – 1991	Fachhochschule Heilbronn **Diplom Betriebswirt (FH)** Schwerpunkte: Wirtschaftsinformatik und Marketing & Sales

Arbeitserfahrung:

Seit 2001	**SAP AG, Walldorf** Weltmarktführer für Betriebswirtschaftliche Anwendungs-systeme **Development Manager im Vorstandsbereich Breakthrough and Innovation: Business Configuration** Start bei SAP als Projektleiter für neue Client Technologien mit Fokus auf Portalanwendungen in der Industrieentwicklung für Energieversorgung, Telekommunikation, Medien und Dienstleistungen. Nach erfolgreicher Produktentwicklung von CRM 4.0 Portalanwendung für Energieversorger fand ein Wechsel in die Strategische Forschung mit Schwerpunkt Konfigurationswerkzeuge und Methoden statt. 2001/2002 Mitarbeit in der DMT (development management team) Arbeitsgruppe Web services, um die Bedeutung von Web services für die Strategische Entwicklung der SAP AG zu untersuchen.

1998 – 2001	**Quark Deutschland GmbH, Ludwigsburg** Quark ist Weltmarktführer für Publishing Software. Durch Produktdiversifikation hat sich Quark entschieden eine e-Business Standard Applikation zu entwickeln mit Schwerpunkt auf der Content-Management and Content-Creation Integration. **Development Manager für die Entwicklung der Business Logik einer Enterprise CRM Applikation** Direkte Führungsverantwortung für 22 Mitarbeiter. Drei Technical Manager im direkten Berichtsweg. Fachliche Verantwortung des Gesamtprojektes bezüglich Design und Implementierung der Business Funktionen und Integration mit anderen Projekten der Quark Produktfamilie. Projektgröße: 150 Mitarbeiter verteilt auf drei Standorte (BRD, Indien, Singapur). Eigenverantwortliche Einstellung der Mitarbeiter am Standort LB gemäß Budgetvereinbarung.
1995 – 1997	**GWI Research, Trier** GWI ist ein Hersteller und Dienstleister für Krankenhausinformationssysteme. **Systemdesign und Anwendungsentwicklung für Anlagenbuchhaltung und Anlagentechnik** Zuletzt verantwortlich für drei Applikationsentwickler und einen Produkt Management und Kundentraining Mitarbeiter. Das Ergebnis war eine Anlagenbuchhaltung vollständig integriert in die Auftragsbearbeitung, Materialwirtschaft und Kostenrechnung mit einem Interface für automatische Buchungen zur Finanzbuchhaltung. Ein Tool für die komfortable Datenübernahme aus bestehenden Anlagenbuchhaltungssystemen war ebenfalls Teil des Leistungsspektrums.
1991 – 1993	**Alcatel Network Services (ANS), BRD** ANS mit Unternehmenszentrale in Paris und Niederlassungen in 5 Ländern Europas sowie eine Niederlassung in den USA ist das Datenkommunikations-Dienstleistungsunternehmen der Alcatel Gruppe. **Marketing and Sales for Value Added Network Services** Value Added Network Services waren teilweise die Vorläufer des E-Commerce.
1981 – 1984	**Raiffeisenbank Heilbronn-Neckargartach** Berufsausbildung zum Bankkaufmann und anschließend ein halbes Jahr Anstellung.
1984 – 1986	Bundeswehreinsatz als Radarflugmelder und Fernmelder in Lauda-Königshofen